Forward Advance Training proudly presents...

CSS3 FOR SUPERHEROES!

Presented by Nicholas Johnson
www.forwardadvance.com

Polyfills

Polyfills are little bits of JavaScript or CSS which backport features into older browsers.

- JavaScript polyfills use a bit of script to add classes, or automatically add styles, IE filters or markup to our page.
- CSS polyfills add fallback styles for a particular browser, either using conditional imports, css cascade order, or JavaScript.
- Filter hacks are IE specific features that allow us to create CSS3 style effects in IE6+

Most polyfills will use a combination of the above

Modernizr

CSS3 PIE IE Sandpaper

IE Filters

You can view the full set of IE filters here:

http://msdn.microsoft.com/en-us/library/ms673539%28v=vs.85%29.aspx

If you thought IE6 was a pain in the bum, wait till you start trying these on. IE filters are unpredictable, buggy, and often just plain weird.

Modernizr, JQuery and YepNope

Modernizr does feature detection, and adds a class to the body for supported attributes. You can combine it with CSS to turn on rules if a feature is detected.

JQuery is everyones favourite DOM manipulation library. Mimic CSS3 effects with dynamically inserted divs and background images.

YepNope is a conditional script loader. Use it in conjunction with Modernizr to load additional libraries when needed.

CSS3 PIE and CSS Sandpaper

CSS3 PIE stands for Progressive Internet Explorer. It works by attaching MS proprietory HTC behavour objects to your elements. It can create rounded corners, shadows and gradients.

IE Sandpaper is a filter hack library. It parses your CSS and inserts IE filter fallbacks. You can get full text rotation and box shadows in IE6, which is pretty extraordinary. Don't push it too hard though, filters are fragile.

Opacity and RGBA

Like a ninja, Opacity allows us to alpha down an element, making it fade away.

RGBA is similar. It allows us to specify a semi transparent colour.

These are different from each other. Opacity sets the opacity of the element and all its descendants, including any text. RGBA colours only affect the colour, so is not inherited.

RGBA

Use it like this:

```
background: rgba(200, 54, 54, 0.5)
```

The four parameters are:

- Red (0–255)
- Green (0–255)
- Blue (0–255)
- Opacity (0–1)

Opacity

Use it like this:

```
opacity:0.5
```

The value is between 0 and 1.

Browser support

Browser support for RGBA and opacity is poor. RGBA works in Firefox, Opera, Webkit and IE9+, also Mobile Safari.

However, and usefully, it works in every browser that supports box-shadow. Box shadows and transparency work very well together.

Making it work in IE

Option 1. A solid colour fallback

Transparency looks nice, but in most cases is not vital to your page, so you may be happy to let IE8 slide. Browsers will ignore the rgba colour if they can't understand it. If you can live without transparency, specify a standard rgb colour first followed by the rgba colour like so:

```
background: rgb(200, 54, 54); /* The Fallback */
background: rgba(200, 54, 54, 0.5);
```

Option 2. A PNG

If you need to support IE7 and 8, you can use a PNG alpha transparency. Create a small alpha'd down square png and specify that as the background right before your rgba rule. This won't work in IE6 because IE6 lacks support for semi transparent PNGs.

```
background: transparent url(pink.png);
background: rgba(200, 54, 54, 0.5);
```

Don't use a single pixel as IE6 may limit the number of images which can be displayed on a page. Instead use a small square, perhaps 10x10.

Option 3. A Gradient Filter

The third option is to use a gradient filter like so:

```
filter:progid:DXImageTransform.Microsoft.gradient(startColorstr=#880000FF,endCol
orstr=#880000FF);
```

The MS Gradient filter accepts an ARGB value. Set each end of the gradient to the same value to get an alpha-ed background. The downside of using a filter is that it disables text smoothing. IE10 removed support for filters.

The first two numbers are the alpha value, ranging from 00 to FF. The next 6 numbers are the RGB value with each pair ranging from 00 to FF.

Transparency Exercise

1. Create a series of overlapping, absolutely positioned divs, and use RGBA values to set their background colours. How pretty, it's like a stained glass window!
2. Add some text to the boxes, try them out. Note how the text is not alpha-ed down. Nice. This will be the case for any nested elements.
3. Now try the same thing with opacity. Alpha it down to 0.1. Notice how the text fades away.

Border Radius

Border Radius allows us to implement rounded corners without using images. It works like this:

```
border-radius: 15px;
```

This will add a little rounded corner to each edge of the element

Particular Corners

If you'd like to specify the corners separately, you can do so using the longhand syntax:

```
border-top-left-radius: 10px;
border-top-right-radius: 10px;
border-bottom-right-radius: 10px;
border-bottom-left-radius: 10px;
```

You can also use the shorthand version:

```
border-radius: 10px 20px 30px 0;
```

These values go round clockwise, starting at the top left.

There's no requirement to make each corner the same radius.

Elliptical Borders

You can also create elliptical borders by providing two parameters like so:

```
border-top-left-radius: 20px 40px;
```

The parameters are:

- horizontal radius
- vertical radius

Eliptical Border Shorthand

You can create eliptical borders using the shorthand syntax. Provide two sets of values separated by a / like so:

```
border-radius: 5px 10px 5px 10px / 10px 5px 10px 5px;
```

The first set are the horizontal radii, the second are the vertical radii.

Browser Support

Browser support is OK, It works in Webkit, Opera, Mozilla and IE9.

If you need to display rounded corners in IE8 you will need to load a conditional stylesheet with background images. There are a variety of JavaScript hacks but these work by adding multiple varying width divs to the page, and add quite a lot of rendering overhead, so I would suggest keeping it simple and just using rounded corner images.

Border Radius Exercise

In these exercises we will experiment with border-radius.

Simple Radii

Attempt to create the following simple effects.

Multiple Radii

Now attempt the following:

BonBon Buttons

Combining Border-radius with box shadow lets us make some very clever effects. Check out BonBon Buttons for an example. We hope for the sake of the internet these do not take off, but the effect is wonderful all the same.

http://lab.simurai.com/css/buttons/

Shadows

CSS3 includes box shadows and text shadows

Box Shadow

Box Shadow can be used to create a drop shadow effect, or an inner glow. Basic usage looks like this:

```
box-shadow: 10px 10px 5px 5px #888;
```

The four parameters are:

- horizontal offset
- vertical offset
- blur (optional)
- spread (optional)
- colour

Multiple Shadows

But box-shadow goes further than this, you can create multiple shadows by passing a comma separated list, like so:

```
box-shadow: 10px 10px #888, -10px -10px #f4f4f4, 0px 0px 5px 5px #cc6600;
```

This allows you to create multiple drop shadows on a single element.

Inner Shadows

You can create an inner shadow or glow using the inset keyword:

```
box-shadow: inset 2px 2px 2px 2px #9ff;
```

This will create a pretty inset shadow or inner glow.

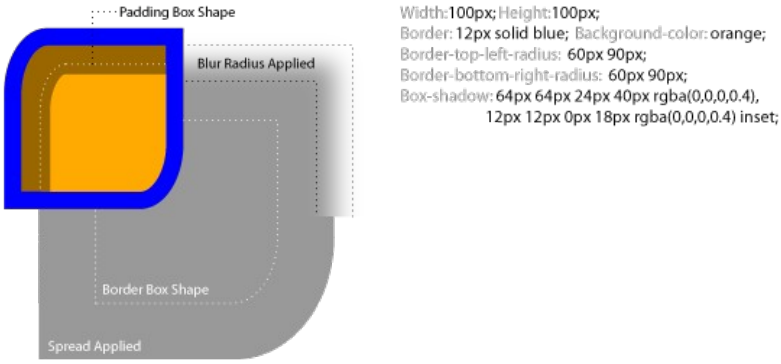
Transparent Shadows

Box shadows are not transparent by default. We can fix this using RGBA values for supporting browsers.

```
box-shadow: inset 2px 2px 2px 2px rgba(0,0,0,0.2);
```

The Spec

The following diagram taken from the W3C specification shows how it should work.



Browser Support

Box shadows are supported in IE9. If you want to polyfill them in older browsers, your best best is to use a background image

At time of writing, vendor prefixes are required.

Text Shadow

Related to Box-shadow is text-shadow. It's not as fully featured as box-shadow, you can't create multiple shadows for instance, but it's still handy to know. Use it like this:

```
text-shadow: #6374AB 20px -12px 2px;
```

The four parameters are:

- Colour
- X distance
- Y Distance
- Blur

The downside, no support in IE9 as of yet. As an enhancement it's fine, but don't rely on it..

Box Shadow Exercise

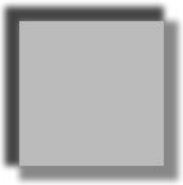
In this exercise we will test out the effects it is possible to achieve using box-shadow.

Using a single box shadow, try to create the following

Using a single box shadow, try to create the following effects. Don’t feel constrained to back and white, please be colourful:

Multiple Shadows

Use multiple shadows to create the following effects



Alpha transparency plus shadows

I have used rgba and 4 shadows to create the following:

1. Try to replicate it.
2. Replicate it using colours, and a wider margin.
3. Add additional shadows, change the position, spread and blur, and create something really colourful.

Inner glow

1. Modify some of your previous examples to use inner glow. See how nice that looks?

Duplicating Elements

One of the great strengths of box shadows is that you can duplicate an element anywhere on the page.

1. Create a circle using a div with rounded corners.
2. Use a shadow with no blur to create a copy the circle off to the right. Give it a different colour.
3. Create several more copies of the circle, each in a different colour.
4. Now do the same with a text shadow. Duplicate a line of text with no blur and a y offset. Do this several times to create a cool effect.

Experiment. See if you can create a fire effect, and ice effect, a blur effect.

See if you can create an RAF target using rounded corners and box shadows.

Selectors

Attribute selectors

We can hang style rules off any arbitrary attribute using attribute selectors. We do this using the square braces syntax. Let's say we have an HTML5 form and some elements are required. We can match them like so:

```
<form>
  <input type="text" name="email" required />
  <input type="text" name="postcode" />
</form

[required] {
  border:1px solid black;
}
```

Equality

We can test for an attribute with a particular value. For example:

```
a[rel="external"]
```

will select every anchor with a rel of external.

Starts with, ends with and contains

We can select within attributes, for example:

Begins with

```
a[href^="https://"]
```

will select every a with an https url.

Ends with

```
a[href$=".com"]
```

…will select every a with an href that that ends with com.

Contains

```
a[href*="www"]
```

…will select every a with an href that that contains www.

Browser Support

Rather wonderfully, support for attribute selectors is pretty good.

IE7+, and all the other browsers support these rules well. If we're happy to drop IE6 we can use these today, and shed a lot of unnecessary classes. No polyfills required.

Structural Pseudoclasses

:first-child

The First Child pseudoclass lets you select the first element in a set.

```
:first_child
```

:nth-child

We can use nth-child to match a particular element or elements in a series, for example:

We can match odd or even elements like this:

```
li:nth-child(odd)
li:nth-child(even)
```

We can match the third object:

```
li:nth-child(3)
```

We can match every third element:

```
li:nth-child(3n)
```

We can also offset every third element like this:

```
li:nth-child(3n+1)
```

Browser Support and Polyfill

Browser support is not so great, with only IE9, plus the other browsers recognising these selectors.

We can polyfill this using a little bit of JQuery. Having imported JQuery, we might do something like this:

```
(function($) {
  $.fn.nth_child_polyfill = function() {
    var i = 1
    this.each(function() {
      $(this).addClass('nth_child_' + i);
      i++;
    });
  }
})(jQuery);

$(function() {
  $('ul li').nth_child_polyfill();
});
```

As you can see, this will add a class to each element. We can then pick up on this class in our stylesheet

```
ul li:nth-child(1),
ul li.nth_child_1 {
    margin-left:0;
}
```

Further Reading

You can view a complete list of selectors plus browser support at quirksmode:

http://www.quirksmode.org/css/contents.html

Selector Exercises

Download the selector exercises markup, now write CSS that matches the following:

1. Every external link (rel equals external)
2. Every google search link (contains google)
3. Every search for cheasypeas (contains cheasypeas)
4. Every UK link (contains .co.uk)
5. The first list item (first-child)
6. Alternate list items (odd or even)

Multiple Backgrounds

CSS3 adds support for multiple background images. We simply supply them in a comma separated list, like so:

```
background: url(top.png) top left no-repeat, green url(bottom) top left, no-repeat;
```

The first image goes on top, the next one goes below.

Note that only the last background can contain a background colour.

Browser Support

Support for multiple backgrounds is not great. It works in Webkit, Opera and Mozilla, but only IE9. A JavaScript fallback is relatively easy to construct for IE8 users.

Partially transparent background images

This brings about a very interesting possibility, that of multiple partially transparent background images.

If we create background images with prime number widths, we can create background images that apparently do not repeat.

We can see some very interesting examples of this in action at the cicada project here:

http://designfestival.com/cicada

Multiple Background Exercise

Create your own cicada backgrounds

Multiple Backgrounds

1. Grab a picture of an animal and a picture of a field from Google Image Search (or use something else, let your imagination run wild).
2. Pop your animal in your field, very nice.

Stripes

1. Using Photoshop, create 4 transparent images of the following dimensions: 10x10, 30x10, 70x30 and 110x110.
2. Fill in a few pixels on each in your own choice of colour, and save as transparent pngs.
3. Stack them up as repeating background images on the body, smallest at the bottom, largest at the top. View in a browser, you should see a pattern which extends some distance without repeating.
4. Add a 17x1 pixel image and lay it on top. Your pattern should now stretch even further.

Squares

1. If you'd like to experiment further, create square images which are multiples of the prime numbers (eg 30x30, 70x70, 110x110, etc).
2. Add your own designs. Use the cicada project to get some ideas

If you tackle this exercise, I'd love to see what you come up with.

Transformation

Transform allows us to perform a variety of transformations to an element. We can rotate it, move it, scale it and skew it. At time of writing, transform requires vendor prefixes.

Vendor Prefixes

CSS3 is an evolving spec, and there's no guarantee that, when a particular new feature is pushed out there, that it will be implemented in the same way cross browser. To counter this we have Vendor prefixes. These are temporary prefixes that namespace particular features.

They are:

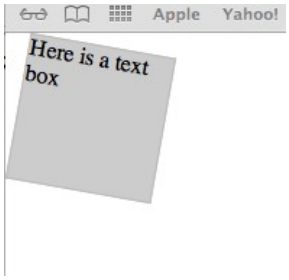
-moz- Mozilla based browsers -webkit- Chrome and Safari -o- Opera -ms- Microsoft Internet Explorer

You'll often see them prepended to edge CSS attributes. As browsers converge around a single implementation we can start to drop them. Until they have converged, you need to include them.

Rotation

Rotate elements like this:

```
-moz-transform: rotate(7deg);
-webkit-transform: rotate(7deg);
-o-transform: rotate(7deg);
-ms-transform: rotate(7deg);
transform: rotate(7deg);
```

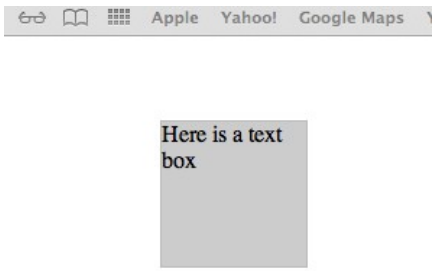


Translation

Translate elements like this:

```
-moz-transform: translate(100px, 50px);
-webkit-transform: translate(100px, 50px);
-o-transform: translate(100px, 50px);
-ms-transform: translate(100px, 50px);
transform: translate(100px, 50px);
```

The first number is the x value, the second, the y value.

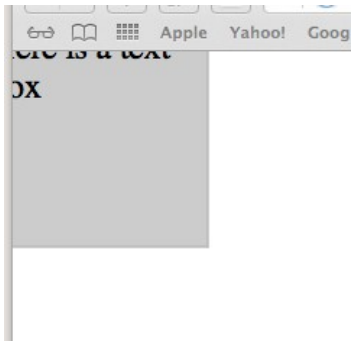


Scale

Scale elements like this:

```
-moz-transform: scale(1.5);
-webkit-transform: scale(1.5);
-o-transform: scale(1.5);
-ms-transform: scale(1.5);
transform: scale(1.5);
```

The value is the amount to scale by, so 1 is no change, 0.5 means half size, and 2 is double size. The whole element and everything inside it is scaled. The scaling starts from the middle, so in this instance we lose quite a lot as the box falls off the edge.

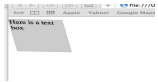


Skew

Skew objects like so:

```
-moz-transform: skew(10deg, 2deg);
-webkit-transform: skew(10deg, 2deg);
-o-transform: skew(10deg, 2deg);
-ms-transform: skew(10deg, 2deg);
transform: skew(10deg, 2deg);
```

The first number is the horizontal skew, so the amount the bottom will have been skewed relative to the top. The second is the vertical skew, the amount the left will have been skewed relative to the right.

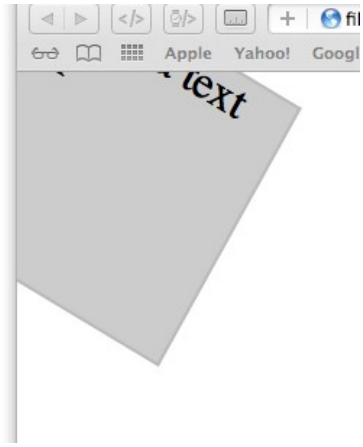


Combinations

We can of course combine transformations together in a single statement like so:

```
-moz-transform: scale(2) rotate(30deg) translate(1px, 1px) skew(1deg, 1deg);
-webkit-transform: scale(2) rotate(30deg) translate(1px, 1px) skew(1deg, 1deg);
-o-transform: scale(2) rotate(30deg) translate(1px, 1px) skew(1deg, 1deg);
-ms-transform: scale(2) rotate(30deg) translate(1px, 1px) skew(1deg, 1deg);
transform: scale(2) rotate(30deg) translate(1px, 1px) skew(1deg, 1deg);
```

Here we have doubled the size of the object, rotated it by 30 degrees, moved it by one pixel and skewed it a little.



Polyfilling

Most of these transformations are beyond the reach of IE8-, with one notable exception, rotation.

IE6 - IE8 include a rotation filter.

```
progid:DXImageTransform.Microsoft.BasicImage(rotation=3);
```

Unfortunately, the filter is quite hard to use, it rotates from the corner instead of the center, and behaves unpredictably when nested inside other elements.

The CSS3 Sandpaper polyfill inspects your stylesheet and writes rotation filters into it. I've used this to good effect on the www.Higgidy.co.uk site.

Transformation Exercise

Let's start out simple:

1. Create a little div. Give it a width and a height.
2. Add a rotation effect
3. Move it a little to the left
4. Rotate it 30 degree left
5. Scale it so we zoom in a little.

A Kaleidoscope

Transformations are inherited. Let's make a kaleidoscope:

1. Create a series of nested divs. Give each a width, height, and RGBA background colour. Now apply a transformation to each div using a rule like this:

   div { -moz-transform: rotate(7deg); }

A messed up site

If you have access to a real site design, you might want to try adding something like:

```
* {
  -moz-transform: rotate(5deg);
}
```

See what you get.

Tricks with forms

The filters work on form elements too.

Modify your first exercise so instead of a div, you apply the translations to a text area. You can type right into the transformation.

Transitions

Transitions provide a way to move smoothly from one state to another. For example, a hover effect could be applied slowly giving a pleasing fade.

Transitions are cut down animations. You can't loop them, or set keyframes. They are simple, easy to apply, and work work nicely.

To make an attribute transition smoothly from one state to another, use the transition property:

```
a:hover {
  -moz-transition:top 2s;
}
```

To make it transition smoothly back again you'll need another transition rule:

```
a {
  -moz-transition:top 2s;
}
```

Transitions don't work with all properties, but do with most of the ones you would care about. A full list of animatable properties (for Firefox) can be found here:

http://www.oli.jp/2010/css-animatable-properties/

Timing Functions

Specify easing using the transition-timing-function attribute.

For example:

```
transition-timing-function: linear;
```

We have the following built in timing functions available to us:

- ease
- linear
- ease-in
- ease-out
- ease-in-out

Triggering Transitions

The simplest, and most common way to trigger a transition is using a hover. You can also trigger one programatically, by adding a class in JavaScript for example.

Polyfills

Transitions are a nice touch, rather than a must have feature. A hover effect will work whether or not it is smoothly animated, so I tend not to bother polyfilling transitions.

If you must though, JQuery includes a pretty good animation library which can be used in place of CSS3 transitions if required. The downside of JavaScript animations is that they

are much more processor intensive. Smoothly animating a single div in JavaScript can easily send the processor to 100%.

Transitions Exercise

1. Create a div. Create transitions on hover.
2. Modify the text size
3. Add a delay, so the transition doesn’t start immediately
4. Try out some timing functions. Refer to the Mozilla documentation for more info
5. Animate a transition, try a rotation and a scale. Subtle
6. Animate -moz-border-radius. This should work in Firefox

An ANimated Kaleidoscope

Use your kaleidoscope from the previous exercise, and add a hover and transition. Points for the best effect.

CSS Art with PseudoElements

Not strictly part of CSS3, the before and after pseudoelements allow you to add up to two additional elements to each normal element.

Pseudoelements are not part of the DOM, and are not visible in the DOM inspector.

Before and After

Sounds like an interior decoration program. These add up to 2 styleable elements to the page.

```
div:before {
  border:1px solid red;
}

div:after {
  border:1px solid red;
}
```

Browser Support

Browser support for pseudoelements is OK, with reasonable support in IE8.

CSS Art Exercises

Triangles Exercise

We can construct a triangle using a little piece of border.

1. Create a div with a width and height of 0px
2. Now set the border width to be 100px
3. Now set the colour of the top, right and bottom sides to transparent. See, we are left with a triangle!

Speech Bubbles

We already know how to make rounded rectangles. Given that we have a triangle, it follows that we can construct a speech bubble.

1. Create a div, give it a width and height, and position relative.
2. Use :after to add another element. Style it as a triangle, and position it absolutely to the top left of it's parent.

Wow, a speech bubble!

Further work

Use your knowledge so far to create a heart, a flower, a cloud, a beetle. Whatever you like. Use transitions to animate your creation. Oooh.

Animation

Animation goes further than simple transitions by adding full keyframe based animation to CSS. This has several advantages, particularly in terms of performance. Because the browser is in charge of the animation it can optimise performance in a variety of ways, taking full advantage of the hardware.

Examples

Check out some examples here:

http://www.webkit.org/blog-files/leaves/

http://www.paulrhayes.com/experiments/clock/

Declaring Keyframes

To make this work, we must first declare the keyframes we want to use, like so:

```
@-moz-keyframes  pop {
    from {
      font-size:100%
    }

    to {
      font-size:500%
    }
}
```

Adding our Animation

Once we have declared our animation, we can add it to the page like so:

```
h1 {
  -moz-animation-duration: 3s;
  -moz-animation-name: pop;
}
```

Here we say that we want our element to pop for 3 seconds, back and forth and carry on forever. Nice.

We can add this as a hover effect if we wish.

Making it loop

We can pop our animation on an infinite loop using the iteration-count property. Here we set it to infinite:

```
h1 {
  -moz-animation-duration: 3s;
  -moz-animation-name: pop;
  -moz-animation-iteration-count: infinite;
}
```

Making it reverse

We can also make it swing both ways using the animation direction property:

```
-moz-animation-direction: alternate;
```

Declaring more keyframes

We can add in additional keyframes using percentages:

@-moz-keyframes pop { from { font-size:100%; } 50% { font-size:1000%; } to { font-size:500%; } }

Here, we have added another keyframe at 50% of the way through the animation.

Animation Exercise

Basic Animation

1. Create a div containing some text
2. Animate its position from left to right. The easiest way to do this is by animating the margin.
3. Add a keyframe for the animation to move through.

Check out the link to the list of animatable properties given in the previous section.

The full documentation is here:

https://developer.mozilla.org/en/CSS/CSS_animations

Cartoon

1. Find several pictures of kittens. Animate their rotation back and forth.
2. Use border-radius to add speech bubbles that move with the kittens.

@Font-Face

It has been possible for a long time to embed fonts on websites. The only problem has been legal. Font owners have been unwilling to allow font embedding as it makes piracy very easy.

We embed fonts on the page using @font-face.

Free Fonts

The best free fonts come from fontsquirrel.com

Visit http://www.fontsquirrel.com and click on @font-face kits. Pick one and download.

Paid for fonts

Paid for fonts tend to be higher quality, and have more swashes and ligatures which will be visible on the latest browsers. There are lots of places to buy fonts. I get mine from www.myfonts.com.

Visit http://www.myfonts.com , and have a look through the bestsellers list. You'll find many you recognise.

Usage

To support a full range of devices you'll need to

```
@font-face {
  font-family: 'Font Name';
  src: url('/fonts/my_font.eot');
  src: url('/fonts/my_font.eot?#iefix') format('embedded-
opentype'),url('/fonts/my_font.woff') format('woff'),url('/fonts/my_font.ttf')
format('truetype');
}

h1 {
  font-family: 'Font Name', arial, helvetica, sans-serif
  font-weight:normal;
}
```

IE Issues

You may be tempted to declare your font as font-weight bold or normal. This is a trap. IE requires that each font has a different name, so your stuck importing bold and regular variant with names like "my-font-bold' or 'my-font-regular'.

When importing fonts, be sure to set any element which would normally be bold to font-weight:normal. If you don't, Firefox will helpfully bold your font up for you. This looks bad.

@Font-Face Exercise

1. Visit Font-Squirel.com, choose and download a font. Very nice.
2. Integrate the Font Squirrel CSS into your page.
3. Style your h1s and h2s.
4. Use a little letter-spacing to make your font look great.

Media Queries

CSS Media Queries are the secret magic that makes responsive design work.

Creating a Conditional Rule

We can include media queries within our stylesheets like so:

```
@media screen and (max-width: 480px) {
  .column {
    float: none;
  }
}
```

Any style rules between the curly braces will be run only if the query passes.

Browser Support

Media queries work in all modern browsers, including the mobile ones, but only IE9.

This might be acceptable, since IE users will be using viewing the site from a desktop machine

Media Queries exercise

Create a web page that responds to the width

1. Add a media query that changes the background colour of the body when the browser drops below 800px.
2. Add another that changes the background when the width goes above 1024

This is the basis of responsive design.

Responsive Design Exercises

Make your page

1. Using Illustrator (or photoshop if you prefer), create a 3 column layout, and make 3 nice square images, each one column wide.
2. Double the size of your images, and save them for web.
3. Create a web page containing an unordered list, and put your images into the list items.
4. Create a float layout. Float each of your list items left and set a width and margin, so together they span the page.
5. Set a max-width on the body, so your page doesn't get too massive.

You should now have a web page that can grow as required, but which looks bad at narrow widths.

Enter the media queries

1. Now create a media query that activates when the viewport moves below 480px.
2. Undo the float layout using float none, and get rid of the gutters by removing the margins.
3. Test your design by changing the background color of the body.

Resize your page and watch it respond. Nice.

Going Further

If you have time left, you may want to extend your design. Here are some ideas.

1. Add a header and a footer.
2. Add a second row of three pictures with a content section in between.
3. Add another layout that triggers at 768 pixels (iPad width).

Feel free to take it further, I look forward to seeing what you make.