The **word-wrap** property

- The **word-wrap** property is to contain overflowing text.

- Problem: It’s not uncommon for people to include URLs in comments and posts, and these URLs often overflow their containers due to their length.

- If the URLs have dashes (-) in them, all the major browsers can wrap the text of the URLs just fine. But Webkit-based browsers and IE will not wrap at the forward-slash (/) character, and none of the major browsers will wrap at underscores (_).
The **word-wrap** Property

- Solution: All you have to do is give the **word-wrap** property a value of **break-word**, and the browser will wrap text within a word if it has to in order to keep it from overflowing.

- The word-wrap property is part of the Text module found at [www.w3.org/TR/css3-text](http://www.w3.org/TR/css3-text). It controls whether or not text is allowed to break within “words.” (The separate text-wrap property controls how lines break between words.)

- The word-wrap property can be set either to **normal** (the default) or **break-word**.
The **word-wrap** Property

```css
blockquote {
  margin: 0 0 0 112px;
  padding: 10px 15px 5px 15px;
  border-top: 1px solid #fff;
  background-color: #A6DADC;
  word-wrap: break-word;
}
```

- The browser will still try to wrap first at normal breakpoints, but if it has to, it will now wrap the text at underscores or even within a word.
The `border-radius` property
The **border-radius** Property

- The **border-radius** property is used to create rounded corners. It is part of the Backgrounds and Borders module found at www.w3.org/TR/css3-background.

- It’s shorthand for the properties specifying the rounding amount of each of the four corners, in this order: border-top-left-radius, border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius.

- You can write out all four values, with spaces in between, in one border-radius property, or just use one value to round all four corners the same amount.
The **border-radius** Property

- You might want to use border-radius for:
  - Buttons; see [http://blogfreakz.com/button/css3-button-tutorials](http://blogfreakz.com/button/css3-button-tutorials) and [http://css-tricks.com/examples/ButtonMaker](http://css-tricks.com/examples/ButtonMaker)
  - Tabs
  - Dialog boxes
  - Circular badges
  - Bar charts; see [www.marcofolio.net/css/animated_wicked_css3_3d_bar_chart.html](http://www.marcofolio.net/css/animated_wicked_css3_3d_bar_chart.html)
The **border-radius** Property

- In CSS3, creating rounded corners can be as simple as `border-radius: 10px` on a single div. No extra markup, no images, no JavaScript.

- We will modify the blockquote rule:

```css
blockquote {
  margin: 0 0 0 112px;
  padding: 10px 15px 5px 15px;
  -moz-border-radius: 20px;
  -webkit-border-radius: 20px;
  border-radius: 20px;
  border-top: 1px solid #fff;
  background-color: #A6DADC;
  word-wrap: break-word;
}
```
The border-radius Property

- The `border-radius: 20px;` declaration specifies that all four corners should be rounded by 20 pixels.

- This syntax is supported by Opera, Chrome, Safari 5, and IE 9.

- Firefox and Safari 4 and earlier use the `-moz-border-radius` and `-webkit-border-radius` properties, respectively.

- With these three lines added, the corners are now rounded in all browsers except IE 8 and earlier. These versions of IE simply ignore the properties and keep the corners straight—no harm done.
Creating Ovals and Circles with border-radius (an aside)

- If you want your speech bubbles to be complete ovals instead of rounded rectangles, you’ll need to use elliptical-shaped corners instead of perfectly round ones. (Elliptical just means that the curve of each corner is somewhat flattened out—just like an oval.)

- To specify an elliptical corner, you write two measurements, separated by a slash, such as this: border-radius: 50px/20px. (Safari 3 and 4 use the non-standard syntax of no slash, just a space.)

- This means that the curve will extend horizontally 50 pixels but vertically only 20 pixels, making a flattened, elliptical curve.

- You can make each corner have different angles; find out how at http://css-tricks.com/snippets/css/rounded-corners.
Creating Ovals and Circles with border-radius (an aside)

- To create circles, first give your box the same width and height; use ems as the unit of measurement instead of pixels to ensure it can grow with its text.

- Then set each corner’s border-radius to one-half the width/height value. For instance, if you have a box that is 10 ems wide and tall, use border-radius: 5em.
Semitransparent Backgrounds with RGBA or HSLA

- One great way to add depth is to make backgrounds semitransparent (also called alpha transparency).

- By letting a little bit of the page background show through, you create more of a layered appearance, as if the semitransparent element is floating over the background.

- Before CSS3, the only way you could create semitransparent backgrounds was by using an alpha-transparent PNG as a tiling background image.
CSS3’s RGBA and HSLA Syntax

- Luckily, in CSS3 we have both RGBA and HSLA to turn to.

- Both are methods for specifying a colour and its level of transparency at the same time.

- RGBA stands for red-green-blue-alpha (for alpha transparency)

- HSLA stands for hue-saturation-lightness-alpha.
CSS3’s RGBA and HSLA Syntax

- So to specify the current shade of blue:
  - Hexadecimal: #A6DADC
  - RGB: 166, 218, 220
  - RGBA: 166, 218, 220, 1
  - HSL: 182, 44%, 76%
  - HSLA: 182, 44%, 76%, 1
CSS3’s RGBA and HSLA Syntax

- In the RGBA syntax, the first three values are the amounts of red, green, and blue, either from 0%–100% or, more commonly, 0–255.

- In the HSLA syntax, the first three values are the hue value, from 0 to 360; the percentage level of saturation; and the percentage level of lightness.

- In both RGBA and HSLA, the fourth value is the opacity level, from 0 (completely transparent) to 1 (completely opaque).
CSS3’s RGBA and HSLA Syntax

- The colour converter tool at [http://serennu.com/colour/hsltorgb.php](http://serennu.com/colour/hsltorgb.php) allows you to convert colour values you already have into hex, RGB, and HSL syntaxes.

- The Doughnut Color Picker at [http://www.workwithcolor.com/hsl-color-picker-01.htm](http://www.workwithcolor.com/hsl-color-picker-01.htm) lets you both pick and convert colours. The picker uses HSL, but gives the hex and RGB equivalents, and lets you input colours in any of the three syntaxes.
Creating Semi-transparent Speech Bubbles

- We currently have a background colour of #A6DADC.

- The HSLA equivalent is hsl(182, 44%, 76%).
  
  ```
  background-color: hsl(182, 44%, 76%);
  ```

- Change to:
  
  ```
  background-color: hsla(182, 44%, 76%, .5);
  ```
Image-free Drop Shadows

- Drop shadows on boxes are created in CSS3 using the `box-shadow` property.

- In the property, you set the shadow’s horizontal and vertical offsets from the box, its colour, and you can optionally set blur radius as well as spread radius.
Image-free Drop Shadows

- Add the following three lines of CSS to the blockquote rule:
  ```css
  -moz-box-shadow: 1px 1px 2px hsla(0,0%,0%,.3);
  -webkit-box-shadow: 1px 1px 2px hsla(0,0%,0%,.3);
  box-shadow: 1px 1px 2px hsla(0,0%,0%,.3);
  ```

- Just as with `border-radius`, all three lines accomplish the same thing, but are read by different browsers
Image-free Drop Shadows

- The first value in each property, 1px, is the horizontal offset from the box, and it tells the browser to move the shadow one pixel to the right of the box’s edge.

- The second value, 1px, is the vertical offset, moving the shadow one pixel down.

- You can use negative values to move the shadow to the left and up instead.
Image-free Drop Shadows

- The third value, 2px, is the blur radius, which specifies over how many pixels the shadow should stretch.

- A larger value makes the shadow blurrier and softer; a value of zero would make it completely sharp-edged.

- The fourth value is the colour—in this case, black at 30 percent opacity.
Image-free Drop Shadows

- You can use any syntax for declaring the colour in box-shadow, but HSLA or RGBA - the only syntaxes that can make a colour semitransparent - are your best bets.

- Semitransparency is very handy for drop shadows, since you want to be able to see the background of whatever is behind the shadow peeking through a bit.
The `text-shadow` property is used to create drop shadows behind text

```
text-shadow: 1px 1px 1px hsla(0,0%,100%,.7);
```

The syntax is almost exactly the same as the syntax for box-shadow. (The only difference is that you can’t set spread radius or inset on text-shadow.)

We have a horizontal offset, vertical offset, optional blur radius, and colour.

In this case, there’s no need to add any browser-specific prefixes; Firefox, Safari, Chrome, and Opera all support the standard text-shadow property
The **text-shadow** Property

- You can apply multiple shadows to the same text by writing each in the same text-shadow property, separated by commas. They’ll be stacked on top of each other, with the first declared shadow on the top.

- The text-shadow property is part of the Text module found at [www.w3.org/TR/css3-text](http://www.w3.org/TR/css3-text). It was part of CSS 2, removed from 2.1, and is back in 3.
The **transform** property

- **Transforms** are a collection of effects, each called a transform function, that manipulate the box in ways like rotating, scaling, and skewing.

- These effects would previously have had to be accomplished with images, Flash, or JavaScript.

- Transforming objects with pure CSS avoids the need for these extra files, once again increasing the efficiency of both your development and the pages themselves.
The **transform** property

- Let's look at the syntax for transforms by rotating an image:

```
img {
  -moz-transform: rotate(-5deg);
  -o-transform: rotate(-5deg);
  -webkit-transform: rotate(-5deg);
  transform: rotate(-5deg);
}
```
Rotating the Avatars

- The transform property (and, for now, all three browser-specific equivalents) tells the browser that you want to apply a transform.

- You then specify that the particular transform function you want is rotate, and that the number of degrees of rotation, using the deg unit, is negative five. You can use either positive or negative values.

- Other transform functions take different types of measurements but the pattern is always the same:

```javascript
transform: function(measurements);
```
The **transform** Property

- The **transform** property is part of both the 3D Transforms module found at [www.w3.org/TR/css3-3d-transforms](http://www.w3.org/TR/css3-3d-transforms), and the 2D Transforms module, at [www.w3.org/TR/css3-2d-transforms](http://www.w3.org/TR/css3-2d-transforms).

- All of the 2D transform functions are also included in the 3D spec, so you may just want to refer to the 3D spec.

- There are too many transform functions to list here, but here’s sample syntax for the most important and supported ones:
The `transform` Property

- **translate**
  - moves the object to a new location, specified as an X and Y coordinate.
  - Positive values move it right and down, respectively, and negative values move it left and up.
  - Example: `translate(20px, -10px)`
The `transform` Property

- **scale**
  - Scales the dimensions of the object X number of times.
  - Negative values flip the object.
  - To scale to something smaller, use a number between 0 and 1.
  - If you use two values, separated by commas, the first is the horizontal scaling factor and the second is the vertical scaling factor.
  - Example: `scale(2.5)` or `scale(1, .5)`
The **transform** Property

- **rotate**
  - Turns an object a specified number of degrees (deg).
  - Positive values turn it clockwise; negative values turn it counter clockwise.
  - Example: `rotate(45deg)`

- **skew**
  - Skews or warps an object, again in degrees.
  - The first value controls the horizontal slope and the second the vertical.
  - If you use only one value, the skew on the Y axis is set to zero.
  - Example: `skew(10deg, 20deg)`
The **transform** Property

- You can include multiple transform functions in one transform property, separated with spaces. The transforms are applied in the order listed.

- You can use the transform-origin property to specify the point of origin from which the transform takes place, using keywords, numbers, or percentages. The default is the centre.

- When you transform an object, the other objects around it don’t move to make way for the transformation (similar to relative positioning). The object is placed first in the flow, and then transformed.
CS7026 – CSS3

CSS3 – backgrounds and fonts
What You’ll Learn…

- The **background-size** property to scale a background image with the text

- Multiple background images on one element

- The **border-image** property to create graphic borders

- The **background-clip** property to move a background image out from under a border

- The **@font-face** rule to embed unique fonts in the page
Scaling the Background Image

- With `background-size`, you can control the horizontal and vertical scaling of a background image as well as how it stretches to cover the background area and gets clipped.
How `background-size` Works

- This is an image 200px wide by 120px tall.

- This is how it looks when set as a normal repeating background of a div that’s 500px wide by 200px tall;

- Since the div’s dimensions aren’t an even multiple of the image’s dimensions, some of the image gets cut off on the right and bottom.
How **background-size** Works

- We can use the `background-size` property to scale the image down from 200 pixels to 100 pixels wide:

```css
#star {
  width: 500px;
  height: 200px;
  border: 1px solid #999;
  background-image: url(images/star.gif);
  background-size: 100px auto;
}
```
How `background-size` Works

- The first value in the `background-size` property, 100px, sets the width of the background image.

- The second value, `auto`, sets the height. A value of `auto` makes the height whatever it needs to be to preserve the aspect ratio of the image.

- If you leave the second value out, the browser assumes it to be `auto` (so a value of `background-size: 100px;` would have worked the same here).
How `background-size` Works

- So the background image has been shrunk but still keeps its aspect ratio:

![Background Image](image.png)

- The browser has scaled the image to 100 pixels wide, so it now fits in the div exactly five times and doesn’t get cut off on the right.
How `background-size` Works

- If you use percentages in the `background-size` property, they’re relative to the box the background is on, not to the background image itself.

- If you wanted exactly two copies of the image to show in the `div`, with neither cut off at all, you could use this CSS:

```css
div {
  width: 500px;
  height: 200px;
  border: 1px solid #999;
  background-image: url(images/stars.gif);
  background-size: 50% 100%;
}
```
How `background-size` Works

- The image is stretched to fill half the width of the div and all of its height, and then repeated.

  ![Image](image.png)

- The browser has to both distort the shape of the image and scale it up, making the edges in the image look a little pixelated.

- As with any browser-based scaling, background sizing doesn’t look good with all images, but can work quite well with grungy, abstract, or very simple images that don’t have super-clean edges.
More New Ways to Tile Backgrounds

- Another way to keep background image tiles from getting cut off on one or more sides is to use the new CSS3 values of **round** and **space** in the **background-repeat** property.

- **round** repeats the background image but rescales it so it will fit an even number of times without getting cut off.

- **space** repeats the background image as often as it will fit without getting cut off, and then spaces the tiles out to fill any leftover room.
Multiple Background Images on One Element

body {
    background: url(images/paperlines.gif) #FBFBF9;
    background: url(images/thumbtack.png),
    url(images/stains1.png),
    url(images/stains2.png),
    url(images/stains3.png),
    url(images/stains4.png),
    url(images/paperlines.gif) #FBFBF9;
    -moz-background-size: auto 1.6em;
    -webkit-background-size: auto 1.6em;
    background-size: auto 1.6em;
}
Multiple Background Images on One Element

- The first background declaration will continue to be used by browsers that don’t support multiple background images.

- Because they don’t understand the syntax of the second background declaration, they’ll ignore it.

- Browsers that do support multiple background images will override the first declaration with the second.
Multiple Background Images on One Element

- The background images are layered on top of each other, with the first declared image put on top of the stack (that’s why the thumbtack image is listed first and the lines image is listed last).

- We still need to tell the browser how we want to repeat, position, and size each image.

- To do this, treat each snippet between the commas as if it were its own standalone background shorthand property, and write each of the background-related properties in it accordingly.
These are all the pieces that can go in the background shorthand property. The order matters for some and not for others, so stick with the order shown:
Multiple Background Images on One Element

- So...

```css
background: url(images/thumbtack.png) 50% 5px no-repeat,
url(images/stains1.png) 90% -20px no-repeat,
url(images/stains2.png) 30% 8% no-repeat,
url(images/stains3.png) 20% 50% no-repeat,
url(images/stains4.png) 40% 60% no-repeat,
url(images/paperlines.gif) #FBFBF9;
```
Multiple Background Images on One Element

- Next, modify the background-size properties to tell the browser that each image should be sized using its native dimensions, except for the last (the lines image):

  - moz-background-size: auto, auto, auto, auto, auto, auto, auto 1.6em;
  - webkit-background-size: auto, auto, auto, auto, auto, auto 1.6em;
  - background-size: auto, auto, auto, auto, auto, auto 1.6em;

- Each comma-separated value matches up with the comma-separated value at the same spot in the background property’s value list.
More on Multiple Background Images

- Multiple background images are a new feature of the background and background-image properties, not a new property itself.

- These properties are part of the Backgrounds and Borders module, found at www.w3.org/TR/css3-background.
Adding a Graphic Border

- CSS3 allows you to assign an image to a border, in addition to (or instead of) a colour and line style.

- The browser will take a single image, slice it into pieces, and stretch or tile each of those pieces across each border.
Adding a Graphic Border

- For instance, let’s say that this is the image we want to use for the borders on a div.

- We want to use the top 30 pixels of the image for the top border, the right 25 pixels for the right border, the bottom 27 pixels for the bottom border, and the left 34 pixels for the left border.

- We need to use these values as both our border widths and our border image slice locations.
Adding a Graphic Border

```
.clouds {
  width: 400px;
  height: 150px;
  border-style: solid;
  border-width: 30px 25px 27px 34px;
  border-image: url(clouds.png) 30 25 27 34 stretch;
}
```

- The first part of the border-image value is the path to the image, which works just like any other path in CSS.

- Next comes one or more numbers to specify where the browser should slice the image. In this case, we’re using four numbers, since we want four different amounts sliced off from each edge.
Adding a Graphic Border

- The first number, 30, is the inward offset from the top edge of the image, in pixels. The second number, 25, is the inward offset from the right edge, the third is the offset from the bottom, and the fourth is the offset from the left.

- The browser will slice the image at each of these lines, creating nine images that it applies to each border, each corner, and the middle of the box.
The Centre Slice

- The spec says that the centre slice should be discarded by default, unless you add the word `fill` to your border-image value.

- However, the older browsers support for the `fill` keyword is patchy with some even filling by default.

- The border-image property is part of the Backgrounds and Borders module, found at [www.w3.org/TR/css3-background](http://www.w3.org/TR/css3-background).
More on the **border-image** Property

- The repeat value can be set to *stretch*, *repeat*, *round*, or *space*.

- Using one repeat value will apply the value to all four sides, while two repeat values applies the first value to the top and bottom borders and the second value to the left and right sides.

- *repeat* will tile all four edges plus the centre;
- *stretch* will stretch them to fill the area;
- *round* will tile and scale them so each fits a whole number of times;
- *space* will tile them so each fits a whole number of times and then evenly distribute the extra space between the tiles.
More on the `border-image` Property

- You need to always set `border-width` in conjunction with `border-image` to create a border area for the image to draw onto.

- You also need to set `border-style` to `solid`.

- There is also a `border-image-width` property, but no browser currently supports it, nor does any browser currently support `border-image-outset`.

- Sadly, border images don’t conform to curved borders created by `border-radius`.
Embedding Unique Fonts

- The `@fontface` rule is a way of linking to fonts on your server (just as you can link to images) that the browser downloads into its cache and uses to style the text on the page.

- It’s often called font embedding (though the fonts aren’t truly embedded anywhere), and the fonts that are “embedded” are called web fonts.
Embedding Unique Fonts

- The `@font-face` rule was actually part of CSS2 back in 1998, but was removed from the CSS2.1 specification. It’s now back, in CSS3, and has widespread browser support.

- The work involved to implement it can be as simple as:

```
@font-face {
  font-family: Raleway;
  src: url(fonts/raleway_thin.otf);
}

h1 {
  font-family: Raleway, Helvetica, Arial, sans-serif;
}
```
Embedding Unique Fonts

- This tells the browser to use the `raleway_thin.otf` font file to render the text inside the `h1` element.

- If the user’s browser doesn’t support `@font-face` or can’t download the file for some reason, the browser simply works through the font stack for a fallback.

- The font stack is the list of fonts declared in the `font-family` property, which the browser tries to load from the user’s machine, in order, until it finds a font it can use.
As you might have suspected, however, using \texttt{@font-face} is more complicated in the real world...
Choosing Acceptable Fonts

- One of the big issues with web fonts is that not every font ought to be used in web pages.

- Some fonts have licensing restrictions that forbid such a use, while others simply don’t look good on the web.
Licensing Issues

- When choosing a font to use, read its license - often called an end-user license agreement (EULA) or terms of use - to see if it allows web font embedding.

- Many fonts’ licenses don’t, because when you use @font-face, the font file is downloaded into the user’s cache, just like images.

- The user could go into her cache, take the font file, and install it on her system.

- Most font vendors are not interested in simply giving their products away to the thousands of people who browse your web site.
Licensing Issues

- Richard Fink describes the problem font vendors have with font embedding in “Web Fonts at the Crossing” ([www.alistapart.com/articles/fonts-at-the-crossing](http://www.alistapart.com/articles/fonts-at-the-crossing)):

  “The fear is that once fonts are on the web, they will become a commodity, the current model will break, and a devaluation of fonts, in general, will occur. The fear is that font designers will no longer be able to charge a print customer, say, $420 for a four-style font family with a 6–10 user license in a world where fonts are being delivered on web sites to virtually unlimited numbers of “users” who don’t have to pay anything at all. What if the web drives down prices in the print sector and doesn’t generate much revenue on its own?”
Licensing Issues

- Unfortunately, most fonts’ licenses were not written with @font-face in mind, so when you read through a font’s license, it may not say anything about not embedding fonts.
- It’s best to err on the side of caution and not use the font unless it explicitly says that web embedding or redistribution is OK.
- This is the case even with free fonts. Just because the font vendor gave you the font for free doesn’t mean you can redistribute it.
- Same thing with the fonts that came with your computer. Again, you have to check the license to be sure.
Font-Embedding Services

- Font-Embedding services offer a collection of fonts that their distributors have approved for web use through the service, getting around the licensing issues of @font-face.

- These fonts are hosted by the service, making them difficult or impossible to download and redistribute.

- Font-embedding services are easy to use because they provide all the different font file formats needed for different browsers, as well as the code for you to add the fonts to your sites.

- This code may include JavaScript in addition to CSS in order to make the real fonts impossible to reuse or speed up their rendering.
Readability & Rendering Issues

- Most commercial fonts were not designed to be viewed at small sizes on a screen, so in many cases it makes the most sense to reserve @font-face for headings and continue to use web-safe fonts like Georgia and Verdana for body copy.

- Every time you choose to use a web font, have a specific reason for picking that font, beyond just that it looks cool.

- Make sure that the font truly enhances the text and doesn’t make it less readable.
Readability & Rendering Issues

- Test your web fonts with your actual content to make sure they will work.

- E.g. the Raleway font below might work well for large headings but be too thin to render well and be readable for body copy.

Always do right. This will gratify some people, and astonish the rest.

- MARK TWAIN
Another aspect of web fonts that can affect legibility is how they are anti-aliased and hinted.

Right now, web fonts are generally more jagged around the edges than traditional fonts, even when anti-aliased, usually because most were not designed to be viewed on screen.

Higher quality fonts, as well as fonts that were designed for the web, have better hinting, which, briefly, is a set of instructions in the font file that adjusts the edges of the characters to line up better with the pixel grids of our computer screens so they look better to the human eye.
Font format plays a role in this too; TrueType fonts are generally better hinted than OpenType CFF fonts.

The degree of jaggedness depends not only on the font but on the operating system and sometimes the browser; Mac is generally smoother than Windows, but can look blurry.
Font Stacks and Readability

- Don’t forget the readability of the fallback fonts in your font stacks.

- Make sure to test the fallback fonts so that if the web font doesn’t load, the user still gets readable and attractive text.

- You usually want to choose fallback fonts that have similar proportions to the web font you’re putting at the front of your font stack.

- That way, the font size, weight, and other styles you apply to the text will work well with whatever font the user sees.
Font Stacks and Readability

Always do right. This will gratify some people, and astonish the rest.
Always do right. This will gratify some people, and astonish the rest.
Always do right. This will gratify some people, and astonish the rest.

- Arial (centre) is ok but Calibri (bottom) is too small to be the best fallback for the Junction (top) web font.

- Trebuchet MS match up well with Junction, with Lucida Sans Unicode being a good runner-up.
Browser Support

- The `@font-face` rule has good browser support, but different browsers want you to use different font file types.

- TrueType (TTF) and OpenType (OTF) font files, such as the ones you probably already have on your computer, work in most browsers.

- IE supports `@font-face` as far back as version 4, but IE 4 through 8 support it only if you use a proprietary font format called Embedded OpenType (EOT).

- EOT is technically not a font format; it’s a compressed copy of a TTF font that uses digital rights management (DRM) to keep the font from being reused.
Using these three formats - TTF or OTF, EOT, and SVG - will make your unique fonts show up in even older browsers that supports @font-face.

However **Web Open Font Format (WOFF)** is the most supported of all font file types. In fact, it is supported by almost all browsers except older versions of Android Browser and older versions of iOS Safari.
WOFF, which stands for Web Open Font Format, was introduced in 2009.

Like EOT, WOFF is not technically a font format, but rather a compressed wrapper for delivering TTF or OTF fonts.

Unlike EOT, however, WOFF contains no DRM.

The WOFF specification became a W3C working draft in July 2010.

WOFF2 provides better compression than WOFF 1.0 but is not yet fully supported.
Browser Support

- So when using custom fonts, it is recommended you install TTF, OTF, EOT, and WOFF font file types in order to cover maximum browser support.

The numbers in the table specifies the first browser version that fully supports the font format.

<table>
<thead>
<tr>
<th>Font format</th>
<th>Edge</th>
<th>Chrome</th>
<th>Firefox</th>
<th>IE</th>
<th>Opera</th>
<th>Safari</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTF/OTF</td>
<td>9.0*</td>
<td>4.0</td>
<td>3.5</td>
<td>3.1</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>WOFF</td>
<td>9.0</td>
<td>5.0</td>
<td>3.6</td>
<td>5.1</td>
<td>11.1</td>
<td></td>
</tr>
<tr>
<td>WOFF2</td>
<td>Not supported</td>
<td>36.0</td>
<td>35.0*</td>
<td>Not supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVG</td>
<td>Not supported</td>
<td>4.0</td>
<td>Not supported</td>
<td>3.2</td>
<td>9.0</td>
<td></td>
</tr>
<tr>
<td>EOT</td>
<td>6.0</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
<td></td>
</tr>
</tbody>
</table>

*IE: The font format only works when set to be "installable".

*Firefox: Not supported by default, but can be enabled (need to set a flag to "true" to use WOFF2).
Converting Fonts

- Some providers of @font-face-ready fonts supply you with all the different font formats you need for the different browsers.

- E.g., Font Squirrel offers something they call “@font-face kits,” each of which includes the original TTF or OTF font, an SVG version, a WOFF version, an EOT version, and a sample style sheet and HTML page showing the @font-face rules you need to place in your CSS (www.fontsquirrel.com).
Converting Fonts

- Even better is Font Squirrel’s @font-face Kit Generator (www.fontsquirrel.com/fontface/generator). You can upload your font and convert it to whichever formats you wish.

- You can also control the CSS syntax it outputs, subset the characters to reduce file size, and use more options to fine-tune the fonts.
Let’s put @font-face to use in our page.

Since it looks like notebook paper, a font that simulates handwriting seems appropriate.

We will use *Prelude*, a casual cursive font, for the headings.

We’re not going to apply a casual cursive font to the body copy, however, as that kind of font at small sizes doesn’t look very good and decreases legibility.
Using @font-face

- We need eight versions of Prelude for our page: EOT, SVG, TTF, and WOFF files for both the regular and bold weight of the font.

- As with almost everything in CSS, there are multiple ways to code @font-face to get the same effect; these different syntaxes use valid, standards-compliant CSS and will work in the same browsers.

- Any of these will work for our purposes, and the choice really boils down to personal preference. We will use the “Bulletproof Smiley” version.
Using `@font-face`

- Here’s what the Bulletproof Smiley syntax for the Prelude font looks like:

```css
@font-face {
  font-family: 'Prelude';
  src: url('fonts/preludeflf-webfont.eot');
  src: local('☺'),
  url('fonts/preludeflf-webfont.woff') format('woff'),
  url('fonts/preludeflf-webfont.ttf') format('truetype'),
  url('fonts/preludeflf-webfont.svg#webfont') format('svg');
}
```
Using @font-face

```css
@font-face {
  font-family: 'Prelude';
  src: url('fonts/preludeflf-bold-webfont.eot');
  src: local('☺'),
  url('fonts/preludeflf-bold-webfont.woff') format('woff'),
  url('fonts/preludeflf-bold-webfont.ttf') format('truetype'),
  url('fonts/preludeflf-bold-webfont.svg#webfont') format('svg');
  font-weight: bold;
}
```
Using @font-face

- These two @font-face rules group the regular and bold font faces into a single font family by declaring them with the same font-family name, **Prelude**.

- Each @font-face rule gives the path to the font files and, optionally, the style characteristics of an individual face (such as font-weight: bold or font-style: italic).
Using @font-face

- Let’s look at just the first @font-face rule for now and go through it line by line.

- **font-family: ‘Prelude’**

  - This assigns a name to the font you’re linking to so that you can later refer to this font in your font stacks.

  - You can make the name whatever you want; it’s just a shorthand way of referring to a whole bunch of font information at once.
Using `@font-face`

- `src: url('fonts/preludef1f-webfont.eot');`
  - This gives the path to the EOT version of the font for IE 8 and earlier.
  - This is separated out from the other versions of the fonts because IE can’t understand a `src` descriptor with multiple comma-separated values.
  - It thinks it’s one big path, preventing it from noticing the EOT and being able to use it when grouped with the other files.
Using @font-face

- The next part of the rule is a second src value that lists all the font files for non-IE browsers.

- Each browser will go through the list until it finds a format it can use, and then download that file, and only that file, to display the text.

- Each font includes a path to the font file, such as `url('fonts/prelude-flf-webfont.woff')`, and a format hint, such as `format('woff')`.

- The format hint is optional, but including it alerts the browsers about the format of each font to keep them from downloading ones they can’t use, which would waste bandwidth and slow page loading.
Using \font-face

- At the start of the second `src` value is `local('😊')`. What in the world does this smiley face do?

- This is there to protect IE. Without it there, IE would try to read the second `src` descriptor as one big path, which would lead it to get a 404 error.

- While this doesn’t stop \font-face from working—IE can still use the separate EOT—it’s an extra, pointless hit on your server that you don’t want.

- IE doesn’t understand the `local('😊')` syntax, and putting it at the start of the `src` value stops it from moving any further into the `src` value, seeing the `url()` value, and then trying to parse the path.
Problems with `local()`

- The `local()` syntax is perfectly valid CSS. Its real purpose in a `@font-face` rule is to point to a locally installed version of the font on the user’s machine, so that if the user has the same font as you’re embedding, he doesn’t have to download the extra file.

- That’s why Paul Irish, who came up with the syntax, recommends using a smiley face: we don’t want to call for a font that might actually exist, and it’s very unlikely that anyone will ever release a font named 😊.
Problems with `local()`

- Letting users skip downloading a font they already have installed sounds like such a good and helpful idea—so why not put the real font name in `local()` instead of a smiley face character?

- This is certainly an option. It’s what Paul Irish’s original “Bulletproof @font-face syntax” did, and you can still choose to download this syntax from the Font Squirrel Generator.
Problems with `local()`

- But before you use the real font name in `local()`, you should be aware of a few problems you might run into:
  - Different fonts sometimes have the same names. It’s possible that the user will end up seeing a completely different font from the one you intend.
  - In previous versions of Chrome, all characters might be displayed as As in boxes if the local font that you’re referring to was installed on the user’s system using the font management software FontExplorer X.
  - In Safari, the user might get a dialog box asking permission to use the local font if it’s being managed by FontExplorer X.
Using `@font-face`

- The second `@font-face` rule declares the bold versions of the Prelude font family. It gives the paths to all the bold font files and also sets the `font-weight` to `bold` inside the rule.

- The font-family name is Prelude (not PreludeBold or some other variation), matching the first `@font-face` rule. Assigning the same name tells the browser that the file is the bold version of the same Prelude font family.

- Now, any time the browser needs to have bold Prelude text (because of a `strong` element in the HTML or `font-weight: bold` in the CSS), it doesn’t have to synthesize the boldness by making the characters thicker, but can instead use the true bold font files.

- Using a true bold or italic font face looks better than having the browser simulate it for you.
Declaring the Font

- Adding @font-face rules to your CSS doesn’t actually make the fonts show up anywhere; it simply links them, so they’re ready to be downloaded and used when you need them.

- Let’s call them up in our h1 and h2 elements. Add Prelude, the name of the font we assigned in the @font-face rule, to the start of the existing font-family values in the h1 and h2 rules.
Declaring the Font

h1 {
  margin: -.3em 0 .14em 0;
  color: #414141;
  font-family: Prelude, Helvetica, “Helvetica Neue”, Arial, sans-serif;
  font-size: 3.5em;
  font-weight: normal;
}

h2 {
  clear: left;
  margin: 0 0 -.14em 0;
  color: #414141;
  font-family: Prelude, Helvetica, “Helvetica Neue”, Arial, sans-serif;
  font-size: 2.17em;
  font-weight: bold;
}
Declaring the Font

- Note that the h1 rule sets the font-weight to normal and the h2 rule sets it to bold.

- This tells the browser to use the regular member of the Prelude font family (the first @font-face rule) for the h1 elements and the bold member of the Prelude font family (the second @font-face rule) for the h2 elements.
Declaring the Font

- We now have handwritten cursive text showing in our headings that is resizeable, selectable, and indexable.
Web Fonts

- Typekit ([http://typekit.com](http://typekit.com)) is a subscription-based service where you pay yearly for access to a collection of fonts, which come from multiple foundries. The smallest collection is free, but has other use restrictions.
- Kernest ([www.kernest.com](http://www.kernest.com))
- [www.fonts.com](http://www.fonts.com)
- Webtype ([www.webtype.com](http://www.webtype.com))
- Typotheque ([www.typotheque.com/webfonts](http://www.typotheque.com/webfonts))
- Just Another Foundry ([http://justanotherfoundry.com/webfonts](http://justanotherfoundry.com/webfonts))
Other useful sites

- The League of Moveable Type (www.theleagueofmovabletype.com) is a small but growing collection of free, open-source fonts that are specifically provided for @font-face use. The Raleway font used is one of these fonts.

- The Webfonts.info site is all about type and typography on the web.

- Font Squirrel (www.fontsquirrel.com) provides a large collection of free fonts whose licenses allow embedding. It also provides some handy tools for working with @font-face.
Places online to find fonts whose licenses allow web font embedding:

- Most of the fonts available at Kernest (www.kernest.com) are free, and all are specifically provided for @font-face use. Some are hosted by Kernest, but most you can download and host yourself.
- exljbris and Fontfabric (http://fontfabric.com) both provide a number of fonts for free that can be embedded on the web, as long as you provide attribution according to the terms in the EULAs.
- All of the fonts at Fonthead (www.fonthead.com) are allowed to be used with @font-face as well as other text replacement methods.
- FontSpring (www.fontspring.com/fontface) sells fonts that can be used both in a traditional way on your computer and in print work, as well as embedded on the web with @font-face.
- FontShop has created web versions of several fonts, called Web FontFonts (www.fontshop.com/fontlist/n/web_fontfonts), that you can buy separately from the traditional versions.
More On Font Hinting And Anti-Aliasing

- “Font Hinting” by Peter Bil’ak (www.typotheque.com/articles/hinting)

Webkit-based browsers let you control the anti-aliasing mode using their proprietary -webkit-font-smoothing property. See “-webkit-font-smoothing” by Tim Van Damme (http://maxvoltar.com/archive/-webkit-font-smoothing) for examples and “Font Smoothing” by Dmitry Fadeyev (www.usabilitypost.com/2010/08/26/font-smoothing) for an argument against the property.
The files Font Squirrel produces are usually all you’ll need, but there are a couple of other tools worth mentioning that will optimize your EOT and SVG files even further.

EOTFAST is free desktop software (download at http://eotfast.com) that converts TTF files into compressed but lossless EOT files; the EOT files that Font Squirrel produces are not compressed.

The command-line tool ttf2svg (http://xmlgraphics.apache.org/batik/tools/font-converter.html) converts TTF files into same size or smaller SVG files; you need to have Java and the Java SVG toolkit Batik installed on your system to run it.