advanced CSS font techniques

@font-face

all code for this lecture is in code snippets

embedding fonts

- Sometimes, you want visitors to see the fonts on your web site exactly as you designed them

- In the past, some designers used images to achieve this effect, thus making the text look exactly as intended
- However, replacing text with images reduces accessibility, because the images can not be read by screen readers and do not resize gracefully

- Using CSS font fallback, a degree of consistency can be achieved across a wide range of devices – so long as the fallback statements are comprehensive (indicate fonts for many devices)

- CSS3 introduced the @font-face rule – this allows designers to embed fonts in a website
- An embedded font becomes available to the browsers during a visit to the site
- Though the rule is still experimental, it is supported by all major desktop browsers, and most mobile browsers have basic support

- Font embedding is not the only intended function of @font-face
    - @font-face can be used to include iconography into your site
    - @font-face can be used to create shorthand font fallback

anatomy of @font-face

```
@font-face
{   font-family: myFontName;
    src: url('Akronim-Regular.ttf');}
```

- Begins the rule

anatomy of @font-face

```
@font-face
{   font-family: myFontName;
    src: url('Akronim-Regular.ttf');}
```

- Defines the name of the font
- This name is used to address the font in other CSS rules in the same sheet

anatomy of @font-face

```
@font-face
{   font-family: myFontName;
    src: url('Akronim-Regular.ttf');}
```

- Beginning of the source declaration

anatomy of @font-face

```
@font-face
{   font-family: myFontName;
    src: url('Akronim-Regular.ttf');}
```

- The location of the font you want to use
- Pathing rules apply – so if your font is located in a different folder in your site, path it properly

example

- To achieve this result we include the following rules in our CSS:

```
@font-face
   {  font-family: myFontName;
      src: url('Akronim-Regular.ttf');}

h1, h2, h3, h4, h5, h6
   {  font-family: myFontName, sans-serif;}
```

acquiring fonts to embed

- Most fonts you have on your PC are owned by a holding company or design company, and licenced for use by your specific operating system or a given software product
- This means that in most cases, you are not legally allowed to embed fonts found on your system

- However, there are many fonts with licences that allow reuse – usually under certain restrictions
- Search for "open source fonts" to find some good sources for fonts you may embed: for example

Google Web Fonts

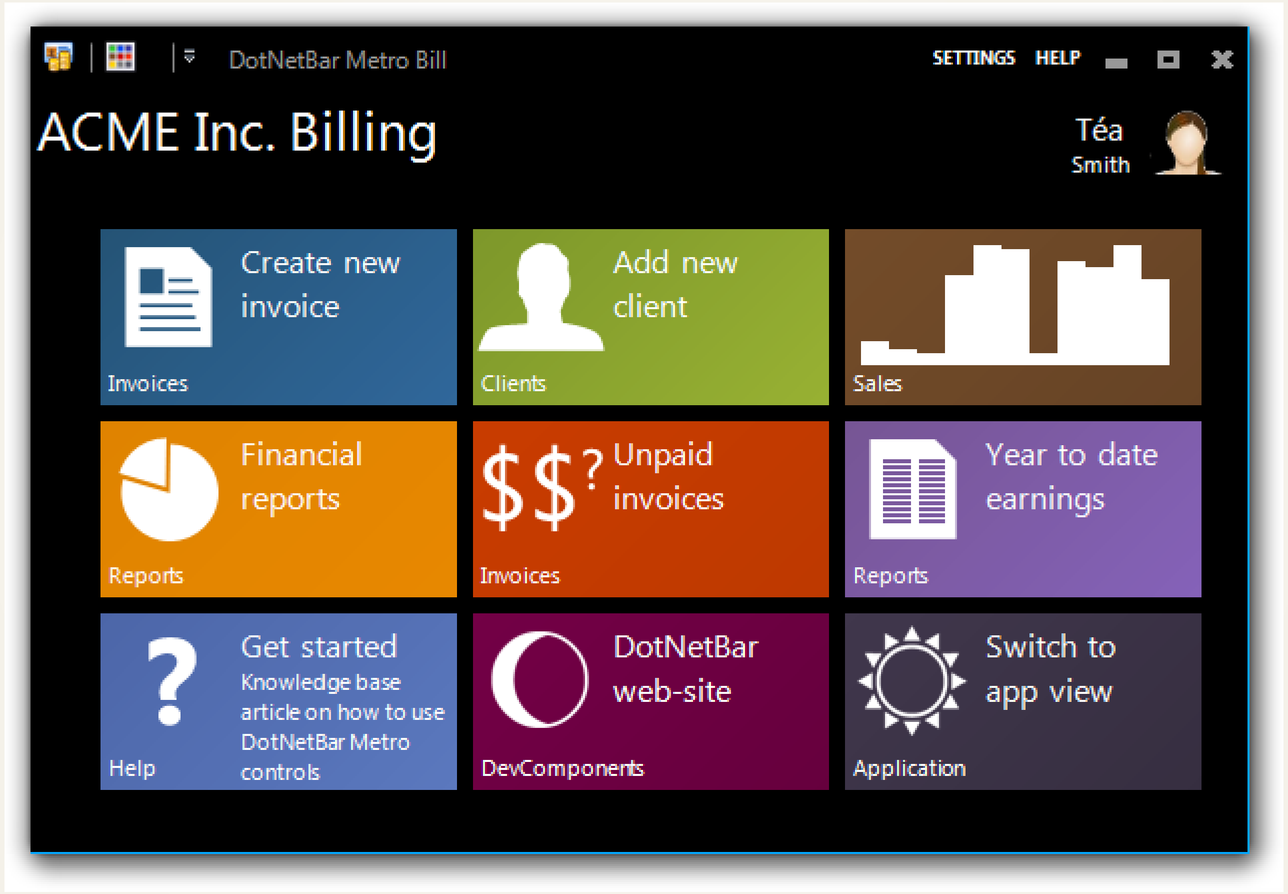
- Google provides a collection of open source, free to use fonts at http://www.google.com/fonts/
- To download any fonts all you need to do is go to the site and:
  1. Find fonts you want to use
  2. Press "Add to Collection" in the bottom right corner of the font preview box
  3. Press "Download your Collection" at the top right corner of the screen

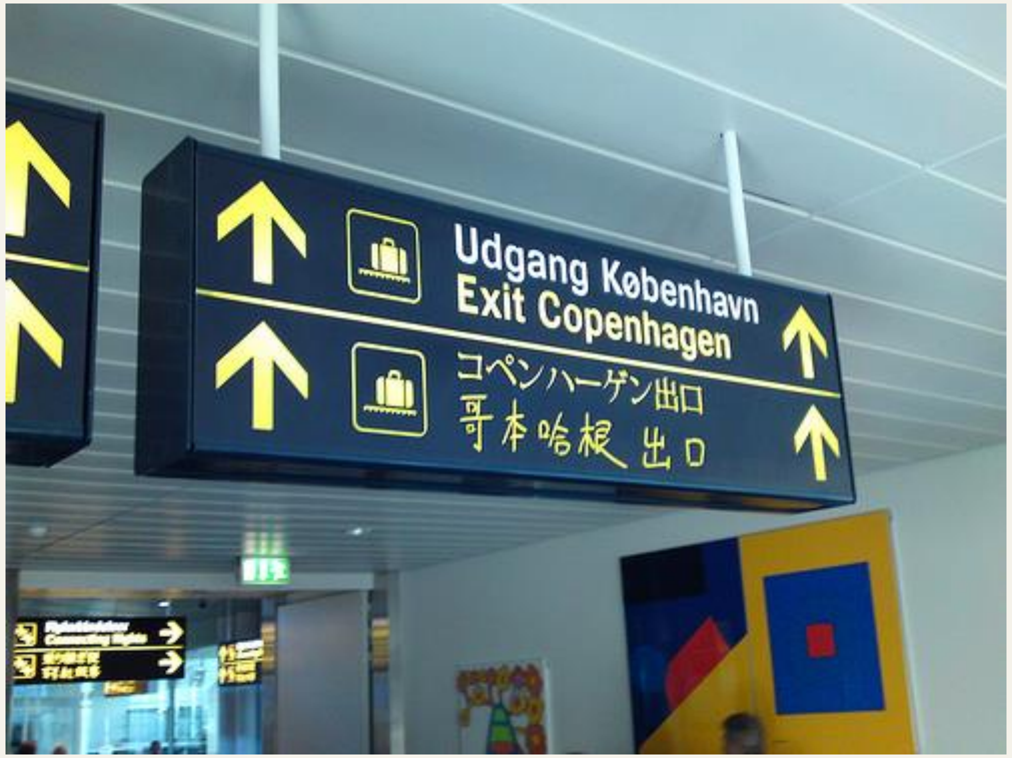
- Google fonts do not require you to download fonts to use them. Pressing the “Use” button at the bottom right will provide with several embedding options

- However, many other sites do not allow external embedding and require you to download fonts, include them in your site, and use the @font-face rule

iconography

- There are many cases where iconography can be more efficient at communicating ideas than words
- You can see examples of informative iconography in airports and subway stations, as well as in modern software design (Microsoft “Metro” style and Google)


DotNetBar Metro Bill
SETTINGS
HELP
ACME Inc. Billing
Téa
Smith
Create new invoice
Invoices
Add new client
Clients
Sales
Financial reports
Reports
Unpaid invoices
Invoices
Year to date earnings
Reports
Get started
Knowledge base article on how to use DotNetBar Metro controls
Help
DotNetBar web-site
DevComponents
Switch to app view
Application


Udgang København
Exit Copenhagen
コペンハーゲン出口
哥本哈根 出口

iconography with images

- Even today, it is common to use images to display icons on web pages
- However images degrade in appearance when resized (example).
- Additionally, images that display well on a computer screen, will not look good on a high density screen (such as the iPhone “retina display”) and vice versa

iconography with fonts

- We can use @font-face rules to import icon fonts to our site (example)

- Since fonts are stored as vector data, icon fonts are infinitely resizable
- Icon fonts can be recolored using the CSS color property – however, like all fonts, any given icon can only be a single color at a given time (example)
- Since icon fonts are text, they can be used in any place text could be – such as an <a> element, or as part of normal text flow

COMPARE

sourcing icon fonts

- IcoMoon provides a free service which generates icon fonts.
- You can select from a set of free icons, pay for access to more, or import your own vector (.svg) files

- .svg vector files can be created and edited using programs such as Adobe Illustrator
- Additionally, many of the illustrations on Wikipedia are .svg files, and are licenced for reuse so long as attribution is provided

IcoMoon

- To create an icon font with IcoMoon, go to http://icomoon.io/app/
- Select the icons you want to include in your font
  - You can find more icons via the "Icon Library" button at the top left
  - You can import your own .svg files by pressing the "import" button at the top left
- Press the "Font" button at the bottom of the screen

- You can rename your font files using the “Preferences” button – this will be the name of the new font-family for this icon font
- Press “Download”
- You will be given a file archive, you need to host these files as part of your website for you to use and the client to see the font

- To use the icon fonts, you will need to merge the provided CSS file with your own, or rename and append it separately
- By default, IcoMoon uses class names to display specific fonts, you will need to append those class names to your elements
- The “index.html” that is bundled in the file archive has a list of class names

- The “style.css” file provided in the archive includes a @font-face rule that resembles the following:

```
@font-face {
  font-family: 'IconFonts';
  src:url('fonts/IconFonts.eot');
  src:url('fonts/IconFonts.eot?#iefix') format('embedded-opentype'),
      url('fonts/IconFonts.woff') format('woff'),
      url('fonts/IconFonts.ttf') format('truetype'),
      url('fonts/IconFonts.svg#IconFonts') format('svg');
  font-weight: normal;
  font-style: normal;}
```

- You may define your own rules that will use the highlighted font-family
- This will allow you to insert icons into your site by using entity names – these are also listed in the “index.html” file

local font instances

- You don’t want to make visitors download a font they already have
- We can check if a client browser has local access to a given font by using the “local” value of src property.
- This should be part of multiple comma separated values of the “src” property

```
@font-face
{  font-family: myFontName;
   src: local('Akronim'),
   url('Akronim-Regular.ttf');}
```

- The url value specifies a path to a font file

```
url('Akronim-Regular.ttf')
```

- The local value should be a font name – the same way that it is declared when part of the “font-family” property – not a file path

```
src: local('Akronim')
```

- Multiple “local” and “url” values may be included as part of the src property of an @font-face rule
- A browser will move through the values one by one from beginning to end, until it finds one of the requested instances of the font – it will then stop

font fallback shorthand

- The way browsers interpret the “src” property of the @font-face rule may be used to create single value shorthand for font fallback
- You will need to create a @font-face rule with multiple local src values, in the same order as if you declare regular font fallback

- The value of the “font-family” property in the @font-face rule can now be used to replace your long fallback strings. This trick can prove very useful when you have multiple elements reusing similar font fallback

example

- The following CSS is used:

```
@font-face
{   font-family: Headings;
    src: local('HelveticaNeue-Light'), local('Helvetica Neue Light'), local('Helvetica Neue'), local('Helvetica'), local('Arial'), local('Lucida Grande');}

h1, h2, h3, h4, h5, h6
{   font-family: Headings, Sans-Serif;}
```

- The "src" property does not accept generic font-families, so the generic font-family still needs to be declared