CS1115/CS5002
Web Development 1

Dr Derek Bridge
School of Computer Science & Information Technology
University College Cork

Web page performance

- Web pages are getting bigger
  - Meadian is nearly 2MB, mostly images
  - See, e.g., HTTP Archive
- Connection speeds are not increasing so quickly
  - Emerging markets mostly use mobile
  - Even in developed regions, mobile is often slow and its use is increasing
- Hence, the web is getting slower
  - Load times of greater than 1 second interrupt the continuity of a user's experience
  - Many sites have average load times of 6 seconds or more; see, e.g., HTTP Archive
  - This results in possible loss of business

Measuring web page performance

- Use the *Network* tab on Chrome's *Developer Tools*
- Software that analyses pages and giving advice:
  - Google's PageSpeed Insights
  - Use the *Audit* tab on Chrome's *Developer Tools*
  - WebPageTest

General advice

- Reduce the number of requests
- Decrease the size of each resource
- Make as much use of caching as possible
- Consider conditional loading or lazy loading
- ...

These may give conflicting advice!

Reduce the number of requests

- Do you really need all those images? fonts? videos? audio files?
- Merge resources
  - Take your external stylesheet and make it an embedded stylesheet
  - Take an image and produce its corresponding **data URI** and then include this in your HTML or CSS
  - Similarly, fonts can be included directly in the CSS
  - Use **image sprites** (background, foreground)
- **Question**: Why might merging be a bad idea?
- Happily, HTTP2 makes merging less relevant to today's Web
  - It still makes separate requests but requests to the same server are multiplexed on the same TCP connection

Data URIs for images

- There are tools for converting images to data URIs, e.g. duri.me
- Here's what the `mojito.jpg` image looks like as a data URI:

```

```

- So now instead of this:

```
<img src="mojito.jpg" title="…" alt="…" />
```

you use this:

```
<img src="
     title="…" alt="…" />
```

Decrease the size of each resource

- Use essential non-redundant HTML, CSS, JavaScript, fonts
- **Minify**, i.e. use a program to remove line breaks, whitespace and comments
  - **Do NOT do this with work you submit to me!**
- Compress resources (or install the `mod_deflate` module and configure your server to compress everything that it serves 'on-the-fly')

Decreasing the size of images

- Choose a format that results in smaller files
  - E.g. PNG rather than GIF (often)
  - E.g. WebP or PNG rather than JPEG (with the problem that WebP is not so widely supported)
- For formats such as JPEG and PNG, increase the compression level as much as you can
- Use a program such as Trimage to reduce crud

Responsive Web Design

- RWD can hurt performance
  - We send the same large image to all devices and then scale it down to fit smaller devices (`max-width: 100%`)
- The `<picture>` element has been added to HTML5 to solve this
  - Create several versions of your image at different dimensions
  - In your HTML, replace the `<img>` element with, e.g.:

```
<picture>
    <source media="(min-width: 70em)" src="mojito_large.jpg" />
    <source media="(min-width: 50em)" src="mojito_med.jpg" />
    <source src="mojito_small.jpg" />
    <img src="mojito_large.jpg" alt="" />
    <p>A mojito is packed with mint leaves and served in a long glass.</p>
</picture>
```

  Note: the media queries are in the opposite order (large to small) to what they were in the CSS (small to large)

  - **Question**: It has a fallback for browsers that don't support `<picture>`. What is it?

Other examples of `<picture>`

- The `srcset` attribute specifies multiple versions of the image with 'hints' to the browser of which to fetch, e.g.

```
<picture>
    <source media="(min-width: 70em)" srcset="mojito_large_1x.jpg 1x, mojito_large_2x.jpg 2x" />
    <source media="(min-width: 50em)" srcset="mojito_med_1x.jpg 1x, mojito_med_2x.jpg 2x" />
    <source srcset="mojito_small_1x.jpg 1x, mojito_med_2x.jpg 2x" />
    <img src="mojito_large.jpg" alt="" />
    <p>A mojito is packed with mint leaves and served in a long glass.</p>
</picture>
```

- The `type` attribute specifies a file format, allowing the browser to fetch the one it supports, e.g.

```
<picture>
    <source media="(min-width: 70em)" src="mojito_large.webp" type="image/webp" />
    <source media="(min-width: 70em)" src="mojito_large.jpg" />
    <source media="(min-width: 50em)" src="mojito_med.webp" type="image/webp" />
    <source media="(min-width: 50em)" src="mojito_med.jpg" />
    <source src="mojito_small.webp" type="image/webp" />
    <source src="mojito_small.jpg" />
    <img src="mojito_large.jpg" alt="" />
    <p>A mojito is packed with mint leaves and served in a long glass.</p>
</picture>
```

Don't forget SVGs!

- SVG images can be
  - either included directly in your HTML/CSS — to minimise requests
  - or stored in a separate file and fetched as an external resource — to allow for caching
- Unless very complex, they tend to be smaller than bitmapped images
- They are just text, allowing for lossless compression
- And they scale — as needed for RWD

Caching

- The simplest solution uses the `mod_expires` module
  - Once installed and enabled (`ExpiresActive` directive), you can then set expiry dates, e.g.:

```
ExpiresActive On

ExpiresDefault "access plus 1 day"
ExpiresByType text/html "access plus 1 week"
ExpiresByType image/gif "access plus 1 month"
```

  - The server will then include the expiry data in the HTTP headers
- There are more complex solutions too!

Conditional loading and lazy loading

- Conditional loading
  - Only fetch resources or parts of resources that the client needs
- Lazy loading
  - Delay fetching non-essential resources or parts of resources until the essential parts of a page are on-screen
- The two are often combined
- For conditional and lazy loading, you need to write some JavaScript
- But soon `img` and `iframe` will have an attribute called `loading` with values such as `eager` and `lazy`