#CI435

Lecture 7

RWD2: Flexible media

Mid-semester Progress Check

- In weeks 8, 9 and 10 we want to check your progress and give you feedback on your work
- On **Friday , November 20th, 15:00** submit your work through the link on the studentcentral **Assessment** page
  - A zip file of your website folder and files - `index.html, tutorial.html` *&c*, css style sheets, image files *etc.*
  - The URL for your website on the Brighton Domains server - *e.g.* http://jh1033.brighton.domains/ci435/index.html
- Your work will not be graded - this does not count towards your final coursework mark
- Verbal feedback will be given by your lab tutor **in class** in weeks in the next three weeks (if you attend)
- You will be given some very brief feedback about your submission online (just to test the process of submi

This lecture will cover…

- Finish last week's slides - *Learning Journal* RWD workflow
- Copyright alert!
- Review RWD – 3 core techniques
- Image basics
- Adding images – HTML or CSS?
- Media in responsive web pages
  - Flexible images
  - Flexible video
- Reading and web resources

CSS - responsive *Learning Journal* page

- The week 6 lab tutorial goes through the **workflow** to make your *Learning Journal* responsive and …
- … lay out the content in a 1, 2 and 3 column CSS grid, depending on the screen width of the device accessing the page
- There is a simple grid template you can use, based on the *Learning Journal* HTML structure - http://jh1033.brighton.domains/ci435/tutorials/learningJournal/grid_template.html
- Media queries lay out HTML elements in 2 and 3 columns depending on screen width of the device accessing the page –

  `@media (min-width: 50em)` – 2 column layout

  `@media (min-width: 60em)` – 3 column layout

Learning Journal RWD workflow

1. I sketched 3 **wireframes** for the layout -
   - Single column mobile
   - 2 column tablet (portrait orientation)
   - 3 column laptop/desktop

http://jh1033.brighton.domains/ci435/tutorials/tutorial06/06wireframes.pdf

These HTML elements are laid out in the **grid** -

```
<header class="banner">
<nav class="menu">
<nav>
<table>
<main>
<aside>
<footer>
```

Learning Journal RWD workflow

- A grid consists of horizontal and vertical lines creating **rows** and **columns** in which content can be laid out
- There are gaps between each row/column ("gutters")
- **CSS Grid Layout** creates the 2 dimensional grid in CSS and places the HTML elements onto it. Elements can span one or more columns and one or more rows.

3. In the style sheet specify a container for the grid: I've used the wrapper element -

```
#wrapper  {
 display: grid;
 grid-gap: 20px;
}
```

This gives a single column grid - *i.e.* the mobile layout

Learning Journal RWD workflow

4. In the style sheet name the areas of the grid using the **grid-area** property -

```
.banner { grid-area: banner; }
.menu { grid-area: menu; }
nav { grid-area: nav; }
table { grid-area: table; }
main { grid-area: main; }
aside { grid-area: aside; }
footer { grid-area: footer; }
```

Learning Journal RWD workflow

5. Set up the grid columns using `grid-template-area` and the `grid-area` names in the same order -

```
#wrapper {
display: grid;
grid-gap: 20px;
grid-template-areas:
"banner"
"menu"
"nav"
"table"
"main"
"aside"
"footer";
}
```

Learning Journal RWD workflow

6. Write a **media query** for a 2 column, 5 row layout -

```
@media (min-width: 750px) {
#wrapper {
grid-template-columns: 2fr 4fr;
grid-template-areas:
"banner       banner"
"menu         menu"
"nav          table"
"aside        main"
"footer       footer"; }
}
```

2 column - width using **fr** unit

"banner", "menu" and "footer" span 2 columns and 1 row

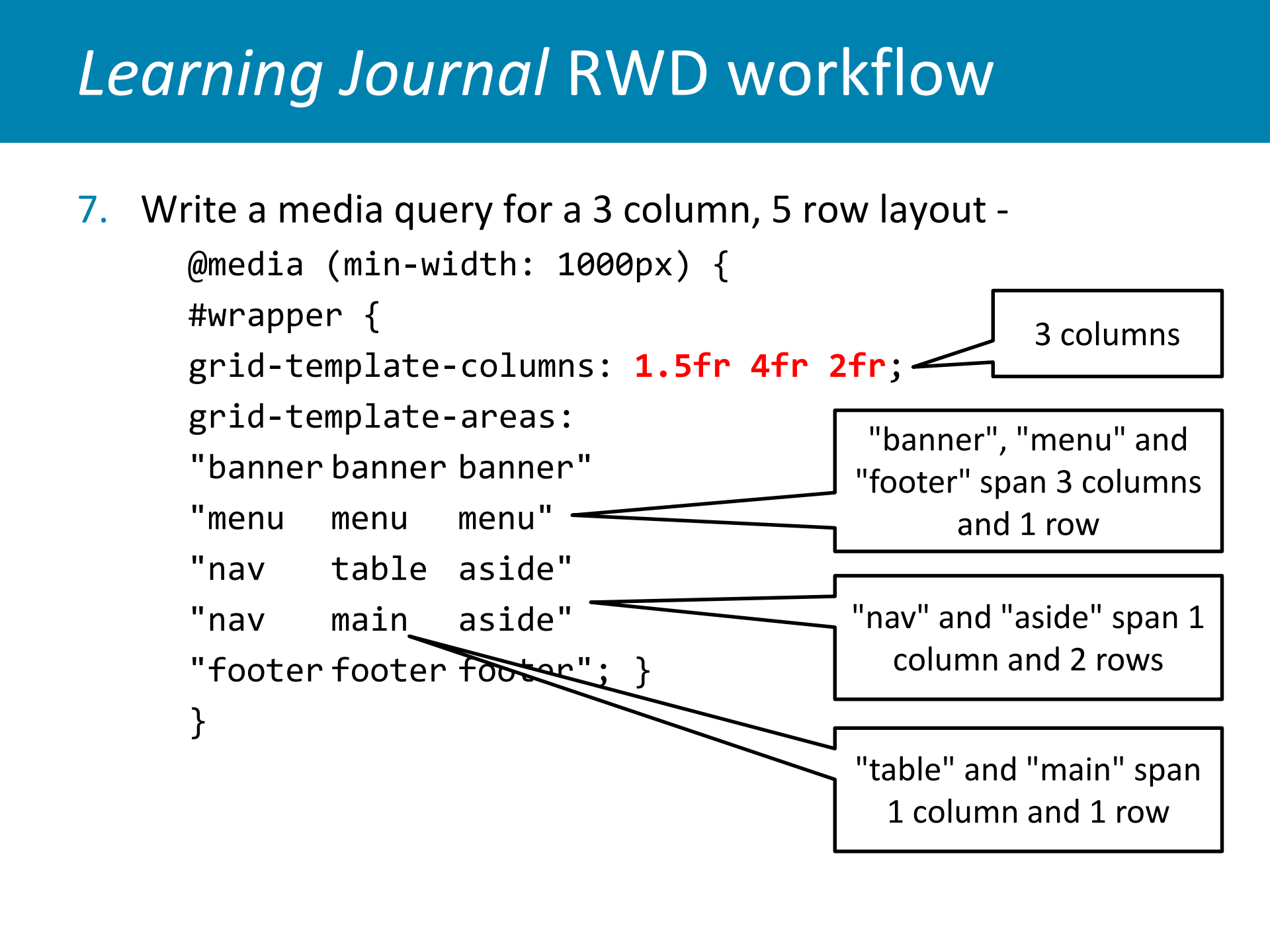
"nav", "table", "aside" and "main" span 1 column and 1 row

- fr is another CSS unit of length. It represents a fraction of the available space in the grid container (#wrapper). It's a flexible unit - content will flow into the available space.

Learning Journal RWD workflow

7. Write a media query for a 3 column, 5 row layout -

```
@media (min-width: 1000px) {
#wrapper {
grid-template-columns: 1.5fr 4fr 2fr;
grid-template-areas:
"banner banner banner"
"menu   menu   menu"
"nav    table  aside"
"nav    main   aside"
"footer footer footer"; }
}
```



Learning Journal RWD workflow

8. Final touch - media query to target very small screen widths and CSS to hide the timetable

```
@media (max-width: 500px) {
        table { display:none; }
}
```

HTML and CSS template -
http://jh1033.brighton.domains/ci435/tutorials/learningJournal/grid_template.html

Fully styled Learning Journal -
http://jh1033.brighton.domains/ci435/tutorials/learningJournal/indexrwd1.html

Learning Journal RWD workflow

Q - Do I have to use a grid to lay out my Learning Journal?

A - No, you decide how to structure and design your pages

- You can pass the coursework without making the website responsive - *i.e.* no media queries - but you won't be able to get a high mark (A or B).
- Media queries style the pages for different devices: *e.g.* could apply CSS to change the menu layout and appearance
- Layout techniques in week 8 can also be used as part of a responsive design solution - flexbox and multi-columns

Lab tutorials

There are TWO online lab tutorials for this week –

1. **Week 7 Lab Tutorial - Flexible Media**: this covers clipped images, flexible images and flexible video containers.
   http://jh1033.brighton.domains/ci435/tutorials/tutorial07/tutorial07.html

2. **Week 7 Extra – Images**: this covers editing and optimising images in Photoshop, CSS background images, favicons and CSS3 alternatives to using image files.
   http://jh1033.brighton.domains/ci435/tutorials/tutorial07/extra-images.html

Copyright alert!

- Can you use copyright images on your website?
  - UK copyright law – “fair dealing” - work can be copied for **private study purposes**
  - You are publishing your work on the web – *i.e.* not private - so you should not use an image or other media created by somebody else without permission
- It's permitted to **embed media** hosted on other sites – *e.g.* videos hosted on YouTube, Vimeo – providing this is not explicitly excluded by the owner
  - YouTube and Vimeo provide embed code and sometimes instructions and text for crediting the source and creator
  - Always credit embedded content correctly

Copyright alert!

- Only use images that are copyright free or licensed for non-profit use – from Wikimedia Commons, or other free stock image libraries and search tools *e.g.* http://compfight.com/
- Look out for media with Creative Commons licenses
- Create original images for the *Tutorial* page
- Reference all your media [and CODE] sources in your *Learning Journal*
  - It's easy for us to find the source of illegal copyright images using Google or TinEye reverse image search
  - Coursework marking criterion - **Awareness of legal and ethical issues** (10% assessment weighting)

RWD – 3 techniques to learn

There are **3 core RWD techniques**

1. **Fluid layout**
   - Use relative units to size elements – **%**, em, rem, fr
2. **CSS media queries**
   - Detect media, viewport sizes, screen widths *etc.*
   - Style elements for devices with different capabilities
3. **Responsive media**
   - Flexible images for different screen widths
   - Flexible media containers *e.g.* video
   - [New W3C standards for responsive images]

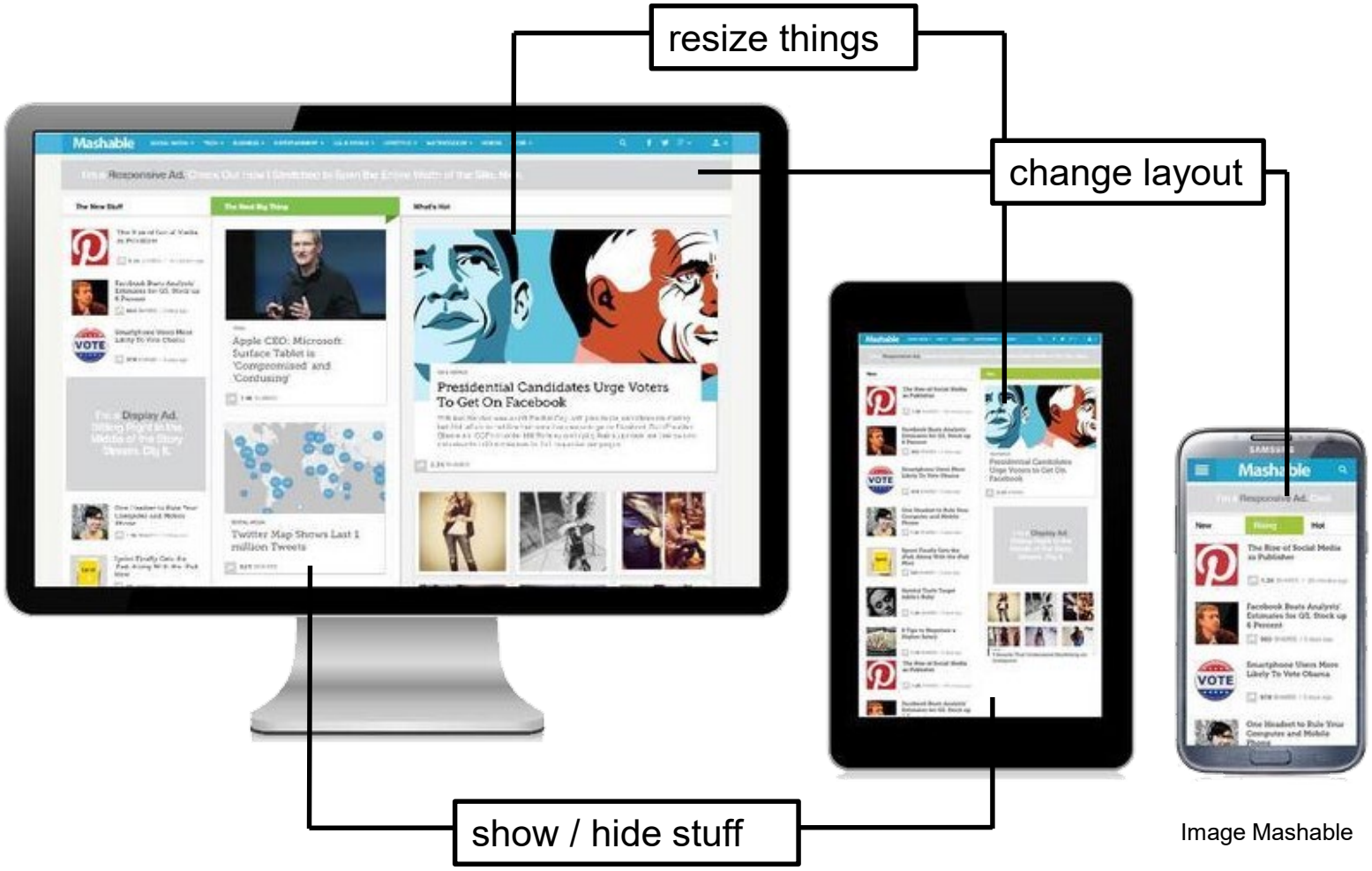


Image Mashable

Example web page for this week and next - http://jh1033.brighton.domains/ci435/tutorials/learningJournal/indexrwd1.html

Image basics

Image and screen resolution

- Digital images on the web are displayed on screens, which consist of a grid of pixels *i.e.* points of coloured light
- So ... both **vector** (SVG) and **bitmap** (PNG, JPG, GIF) images are displayed as pixels on a computer device screen
- **Screen resolution** refers to the number of pixels displayed per unit of length on the screen
- Usually measured as **pixels per inch – ppi (dpi)**
- Screen resolution depends on the size of the screen (number of pixels that can be displayed) plus its ppi setting
- Read Peterson, C., *Learning Responsive Web Design*. Chapter 6: Images

Image and file size

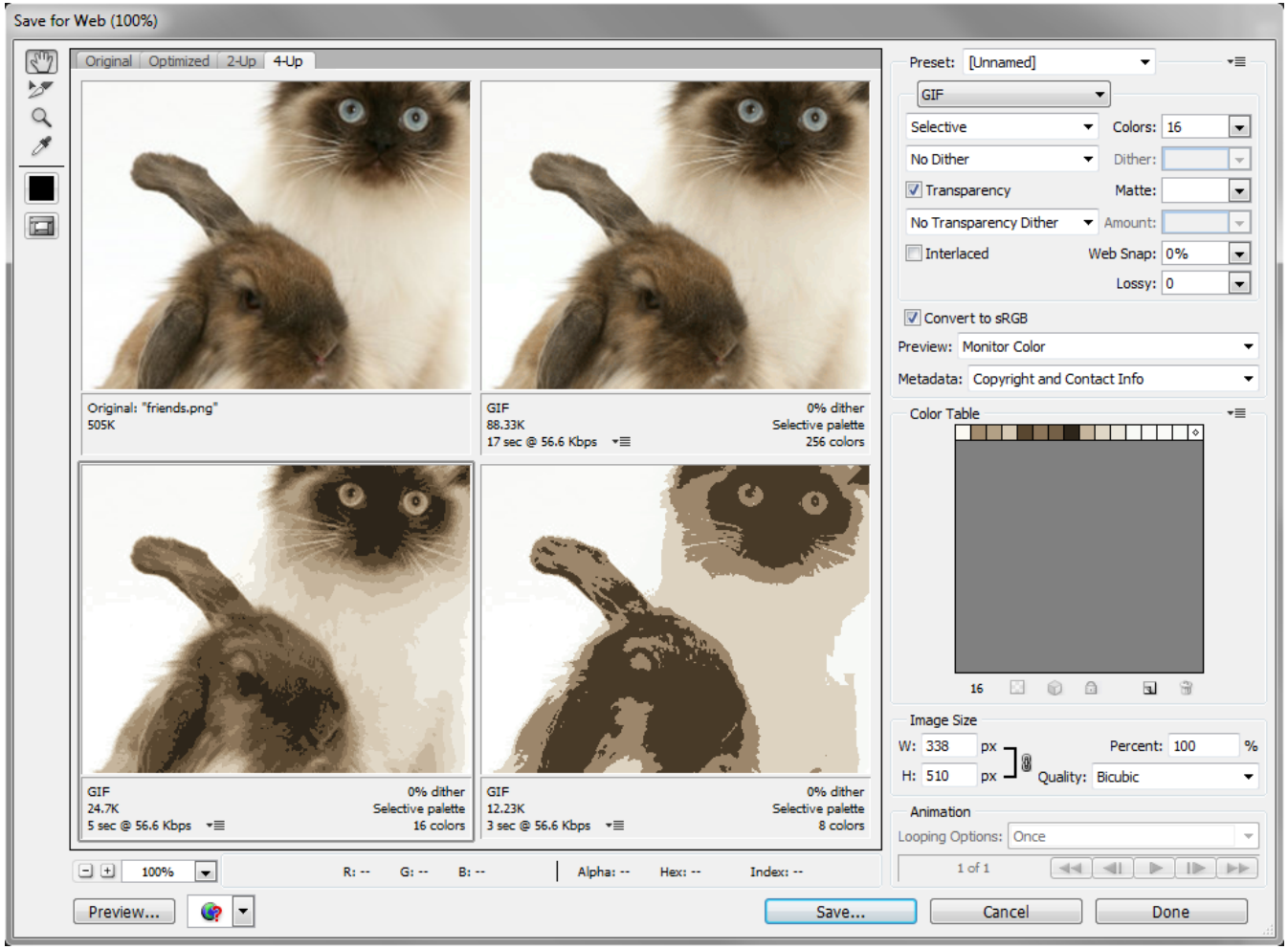
Size of an image –

- Image **resolution** = the number of pixels in a linear inch of an image's width (W) x height (H): measured in **dots per inch – dpi**
- The **pixel dimensions** of an image = the number of pixels along its height and width *i.e.* its size in pixels
- **File size** is the size of an image file measured in kilobytes (K), megabytes (M) or gigabytes (G)
- File size and image quality increase in direct proportion to the resolution and pixel dimensions of an image

Image quality versus file size

- **File size** is an issue – large images load slowly when viewed online and compromise web page usability
  - Particularly an issue for users of mobiles - data download limits and may access web over a slow connection
  - Google takes loading speed into account when indexing
- Images for the web can be **optimised** to get the best trade-off between file size and image quality – save images at the **RIGHT DIMENSIONS** , **RESOLUTION** and **ACCEPTABLE QUALITY**

Photoshop - optimising images tool



Compare different file formats, sizes and quality before saving the image – **trade off quality and file size**

Click File>Export>Save for Web (Legacy) Alt+Shift+Ctrl+S

MS Paint – comparing file size

png

gif

jpg

tif

Name	Size	Type
cat.tif	1,344 KB	TIF File
cat-ORIGINAL.png	1,310 KB	PNG File
cat.gif	347 KB	GIF File
cat.jpg	254 KB	JPG File

NOTE: file size is different to display size

Re-sizing images

- All images used on your web pages must be edited to re-size them and optimised for the web
- This is important if you are using images taken by a digital camera because these are -
  - High resolution
  - Large pixel dimension
  - Very large file size
  - Slow to load
- We can check to ensure you have resized your images – http://jh1033.brighton.domains/ci435/tutorials/resize-images.html

Images - HTML or CSS?

- If the image is **informative content** it should be in the HTML document *e.g.* icons, photos accompanying text
- Always include a text description for the **`alt` attribute**
- This makes the image **content** accessible – to screen readers, text only browsers, search engines

```
<img src="images/HTML5logo.png" width="94"
height="110" alt="HTML5 logo">
```

- `Width` and `height` measurements are the actual pixel dimensions of the image - don't make the browser re-size the image
- **Don't use `width` and `height` attributes if you need flexible images**

Images - HTML or CSS?

- **Presentational** images can be added as background images in css stylesheet -

```
.banner {
  background: url(images/banner.png) no-repeat;
  width: 100%; }
```

  - If the containing element for the background image is fluid (%) this image will also be fluid and resize
  - Background images cannot be given descriptive text and are therefore not accessible
  - One advantage is that text content - *e.g.* a page heading - can be written in the HTML. This can be indexed by Google.

Flexible image techniques

Flexible images

- Bitmap images have fixed pixel dimensions – **height** and **width**
- This is a problem when making fluid and responsive pages – *i.e.* layouts measured in percentages that resize in the browser viewport
- Pixel-sized images may overflow their containing elements
- **Flexible images** – measured **as a percentage of the element in which they are nested** – are the first step in RWD
- Images should be optimised for the **maximum** size they will display – *don't* resize the image beyond this point in the browser as its quality will degrade
- Keep file sizes down – the browser will be scaling images for different screen widths and this can effect performance

Avoid using bitmap images

CSS3 has added many properties that provide a natively responsive alternative to using bitmap images *e.g.*

- `border-radius` (rounded corners on elements)
- `border-image` (image in a border)
- `box-shadow` (drop shadow on boxes)
- `text-shadow` (drop shadow on text)
- `linear-gradient, radial-gradient, repeating-linear-gradient, repeating-radial-gradient`
- `rgba` and `hsla` color values (transparency – alpha channel)
- The CSS Shapes module – for defining the box as a geometric shape https://css-tricks.com/examples/ShapesOfCSS/

Avoid using bitmap images

- SVG – Scalable Vector Graphics – an XML-based language where the characteristics of the images are described - *e.g.* path, coordinates, colour
- Unlike bitmap images, if the size of an SVG graphic is increased the data remains the same, so does the file size
- Use SVG for simple graphics that can be described as paths – ideal for icons
- Not suitable for photographic images with a big colour range or gradients

Flexible images

- Use **flexible images** that rescale as the layout resizes

  CSS – `img {max-width: 100%}`

  http://jh1033.brighton.domains/ci435/tutorials/tutorial07/example.html
- Optimise images for the **maximum width** they will display, because when bitmap images are enlarged pixellation occurs
- Defining `max-width` for a container element in which all elements are nested helps to calculate this parameter
- This solution means that a larger than needed image file is displayed on small screen devices - scaled down by the browser
- Keep file sizes down by optimising images

Traditional techniques - clipping images

- To work with fluid layouts, the image can be clipped when the page is re-sized in the browser
- **'Art direction'** - image should be designed appropriately, so that essential information is not lost

```
.banner {
overflow: hidden;
}
```

http://jh1033.brighton.domains/ci435/tutorials/tutorial07/example.html

Flexible images - %

To make an image flexible -

html

```
<header class="banner"><img src="banner.png"
alt="website banner" /></header>
```

css

```
.banner img  { max-width: 100%; }
```

- The image will never exceed 100% of its parent element and will adapt to the device viewport size
- No need to declare the `height` value – the browser will scale the image proportionally keeping the same aspect ratio

  http://jh1033.brighton.domains/ci435/tutorials/tutorial07/example.html

Flexible images - %

- Declare the **`max-width`** property to prevent the image from resizing beyond its maximum **actual pixel width** and becoming pixelated

```
.banner img  {
    width: 100%;
    max-width: 1000px; /*actual pixel width
                         of the image*/
    }
```

Flexible images - %

- In this fluid layout example, the image is floated; measured as 33% of the `<article>` element in which it is nested -

```
article img {
  float: left;
  padding: 1em;
  width: 33%;
  max-width: 30em; /*Actual pixel width = 480px*/
  min-width: 18.75em; /*18.75em = 300px*/
}
```

http://jh1033.brighton.domains/ci435/tutorials/tutorial07/example.html

Flexible media – embedded video

Flexible media - video

- Embedded video presents a challenge for responsive design
- Ways to include video on a website –
  - Self-host the video and include it in content using the HTML5 **<video>** tag
  - Host video on YouTube or Vimeo, which provide code to embed videos in html in an **<iframe>**

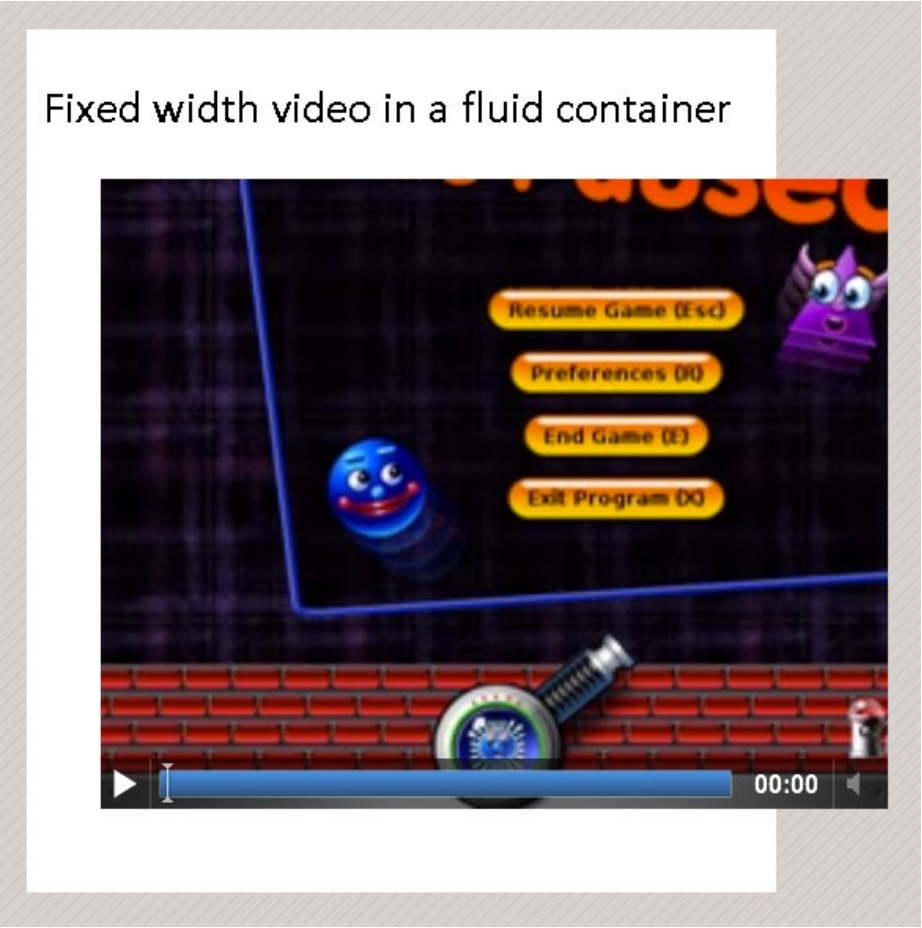
```
<iframe width="560" height="315"
src=https://www.youtube.com/embed/TrduBSGtkS4
frameborder="0" allowfullscreen></iframe>
```

  - Note that pixel **width** and **height** are declared - **aspect ratio** of video is **16:9**
  - If you want a frame border define it in CSS, not html

Flexible media - video

- A static width video will break out of a fluid width container element when it's resized
- **HTML5 video** can be made flexible when styled by the same `css` as we used for `<img>` –

```
video {
width: 100%;
height: auto; }
```



Flexible media - video

- Unfortunately the `css` solution does not work for embedded video delivered through an `<iframe>` -

```
iframe {
  width: 100%;
  height: auto; }
```

- The video displays full width, but the height is displayed at 150px, which is the browser default height for `<iframe>`, `<embed>`, `<object>` and `<canvas>` elements

http://jh1033.brighton.domains/ci435/tutorials/tutorial07/example.html

Flexible media – video: HTML

- We need to declare the video height, but in a way that maintains the 16:9 aspect ratio of the video
- A flexible video height must be a measurement that is relative to the video width as it is resized *i.e.* **56.25%** (or **9/16ths**) of the width
- The solution is to nest the video `<iframe>` in a container `<div>`, then use CSS to give it an **intrinsic aspect ratio** and then **absolutely position** the `<iframe>` within that
- The HTML markup -

```
<div class="video-container">
    <iframe width="560" height="315"
    src=https://www.youtube.com/embed/TrduBSGtkS4
    allowfullscreen></iframe>
</div>
```

Flexible media – video: CSS

```
iframe      {
    width: 100%;
    height: auto; }
.video-container {
    position: relative;
    padding-bottom: 56.25%; /*16:9 =9/16 of the width*/
    height: 0; }
.video-container iframe {
    position: absolute;
    top: 0; left: 0; right: 0; bottom: 0;
    width: 100%;
    height: 100%;}
```

http://jh1033.brighton.domains/ci435/tutorials/tutorial07/example.html

This week's reading

- Duckett, J., 2011. *HTML & CSS: design and build websites.* Read Chapter 5 - Images
- **Peterson, C., 2014. *Learning Responsive Web Design.* Chapter 6 - Images.**
- Marcotte, E., 2011. Fluid images. *A List Apart.* http://www.alistapart.com/articles/fluid-images/
- If you want to go further and read about responsive images - http://www.webdesignerdepot.com/2015/08/the-state-of-responsive-images/