CS1115/CS5002

Web Development 1

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

Design-for-all

- We have looked at **accessibility** (previous lecture)
- Here we look at two more problems:
  - Visitors use different web browsers. How do we ensure **cross-browser compatibility**?
  - Visitors use different devices (differing especially in terms of screen dimensions). We see how *Responsive Web Design* solves this

Cross-browser compatibility

- There are numerous browsers and many different versions of each browser
  - Chrome, Firefox, Internet Explorer, Safari, Opera, Edge,...
  - Chrome for Android, iOS Safari, UC Browser for Android, Opera Mini, Android Browser, Opera Mobile, Firefox for Android,...
- The 'browser wars' are long over and everyone now agrees that we use web standards
- So why might one browser differ from another?
- HTML5 and CSS3 are constantly being improved
  - Older browser versions may not support the latest features
  - Older browsers may have bugs in their handling of CSS
  - Some features are not finalised and yet the newest browsers may have experimental support for them
- Useful resources:
  - caniuse.com
  - html5please.com/

What should you do?

- First, find out what browsers your visitors use by analyzing your server **access log files**
  - For Apache, `mod_log_config` must be installed and enabled
- Second, see what your web site looks like in the browsers that your visitors most often use
  - Q: Do web sites need to look exactly the same in every browser?
  - Q: If not, then what should you try to achieve?
- Third, use the following to overcome any problems:
  - Least-capable-clients-first and progressive enhancement
  - Feature detection
  - Fallbacks (not covered in this module)
  - Polyfills (not covered in this module)

Least-capable-clients-first and progressive enhancement

- Write nice HTML so everyone can access your content
- Add layers of CSS
  - first simple stuff that every browser will understand
  - then, if needed, more and more layers of fancier stuff (ignored by older browsers)
- Similarly with any JavaScript that you need

Different devices

- A few organizations have two web sites, one for 'desktop' and one for 'mobile', (with different URLs), e.g.
  - https://en.wikipedia.org/wiki/Napoleon
  - https://en.m.wikipedia.org/wiki/Napoleon
- Their server may even try to automatically decide which version to serve from headers in your HTTP request
- *Responsive Web Design* (RWD) is a widespread, alternative
  - Adapt the layout to the characteristics of the device

Feature detection

- In the CSS, check whether the browser supports a feature or not

```
@supports not (display: flex) {
    nav {
        width: 25%;
        float: left;
    }

    … /* Other stuff for a non-Flexbox layout goes here */
}

@supports (display: flex) {
    body {
        display: flex;
    }

    … /* Other stuff for a Flexbox layout goes here */
}
```

- But...`@supports` is not supported (!) in early browsers
  Question: What simple fix to the example above would overcome this problem?

Responsive web design (RWD)

- A flexible (liquid) layout, based on a grid
- Flexible images and media
- Media queries
- Least-capable-clients-first ('mobile-first')
- Progressive enhancement

CSS3 Media queries

```
body {
    color: white;
    background-color: black;
}

@media screen and (min-width: 480px) {
    body {
        background-color: red;
    }
}

@media screen and (min-width: 768px) {
    body {
        background-color: blue;
    }
}

@media screen and (min-width: 1024px) {
    body {
        background-color: yellow;
    }
}
```

- `min-width` refers to the viewport (browser window)
- This CSS says:
  - Use a black background
  - If the viewport is at least 480px, use a red background
  - If the viewport is at least 768px, use a blue background
  - If the viewport is at least 1024px, use a yellow background
- **Question**: Suppose the viewport is 1200px, then all the above are true. Why will the background be yellow?

CSS3 Media queries

- Simple media queries comprise
  - the media type: `screen`, `print` and 8 more
  - an expression: a media feature (e.g. `min-width`) and a `value` (e.g. `768px`) in parentheses
  - some CSS between curly braces
- A list of the media features that you can use
- The browser applies the enclosed CSS *only if* the query is true

Media queries for RWD

- Use media queries to adapt to a range of devices:
  - Adapt the overall layout, e.g.
    - a single-column layout for narrower viewports
    - one or more multi-column layouts for wider viewports
  - Adapt individual components, e.g.
    - vertically-stacked navigation menus versus horizontal ones
    - headers containing logos above slogans versus logos next to slogans

  (Sometimes you don't need media queries to make components adaptive — consider how adaptive flexboxes can be)

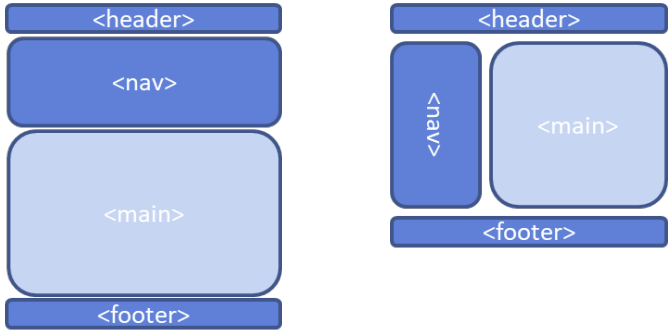
Least-capable-clients-first and progressive enhancement

1. Write nice HTML
   - proper use of HTML markup
   - logical order
   - validate it

   ...for screen readers, search engine crawlers, etc.
2. Write some core CSS
   - something that looks OK on all devices
   - e.g. single-column layout for the narrowest reasonable viewport widths
3. Write successive media queries that apply extra styles for relevant **breakpoints**
   - e.g. multi-column when viewport `min-width` allows it
   - e.g. even ultimately a fixed-width design for very wide viewports

Simple example

- In narrow-screen devices, we want one-column layout
- In wider devices, we want two-column layout



Simple example: CSS

```css
/* Core */

body {
    width: 80%;
    margin: auto;
    font-size: 16px;
}

/* Any more core CSS goes here */

@media screen and (min-width: 50em) {

    body {
        display: grid;
        grid-template-columns: 75% 25%;
        grid-template-rows: auto;
        grid-template-areas: "upper-top upper-top"
                             "lower-top lower-top"
                             "middle-left middle-right"
                             "bottom bottom"
    }

    header {
        grid-area: upper-top;
    }

    nav {
        grid-area: lower-top;
    }

    main {
        grid-area: middle-left;
    }

    aside {
        grid-area: middle-right;
    }

    footer {
        grid-area: bottom;
    }

}
```