#CI435

Lecture 6

RWD1: Introduction to responsive web design

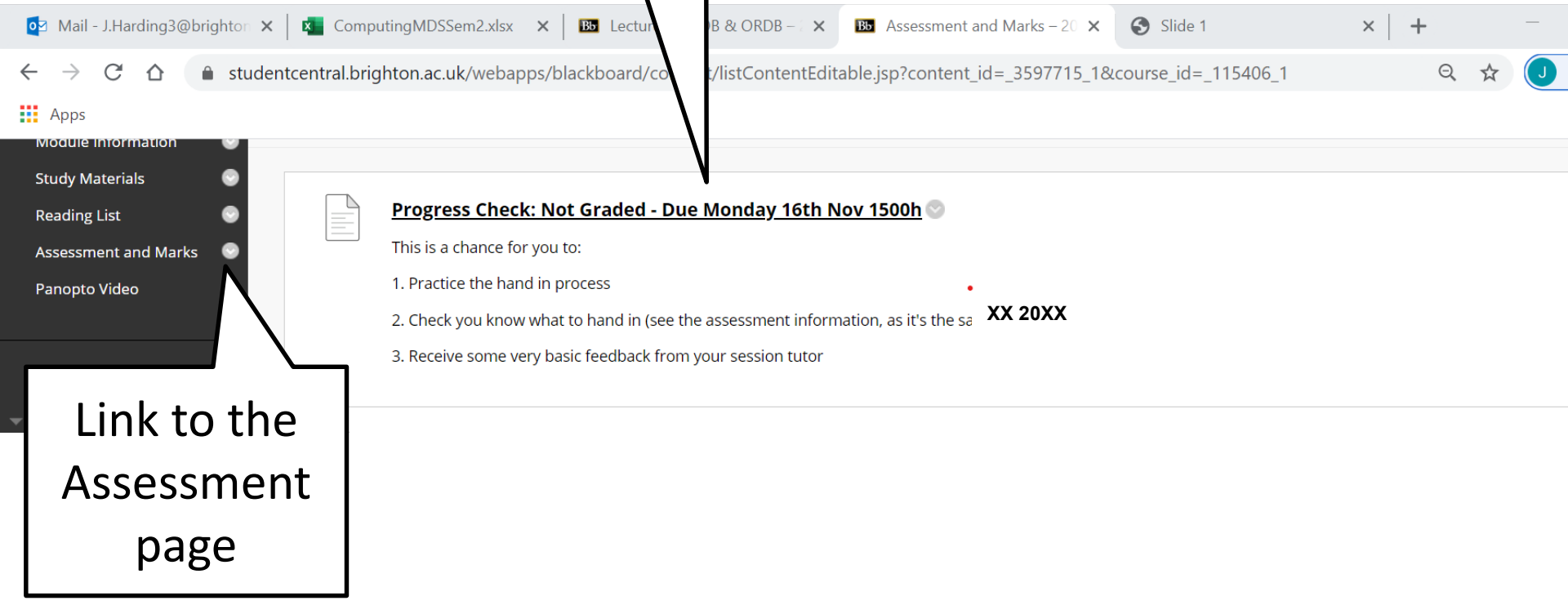
Mid-semester Progress Check

- In weeks 8, 9 and 10 we want to check your progress and give you feedback on your work (and practice hand-in)
- By **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
- Feedback will be given by using the rubric that you can see in the Assessment and Marks area. More feedback will be available in class

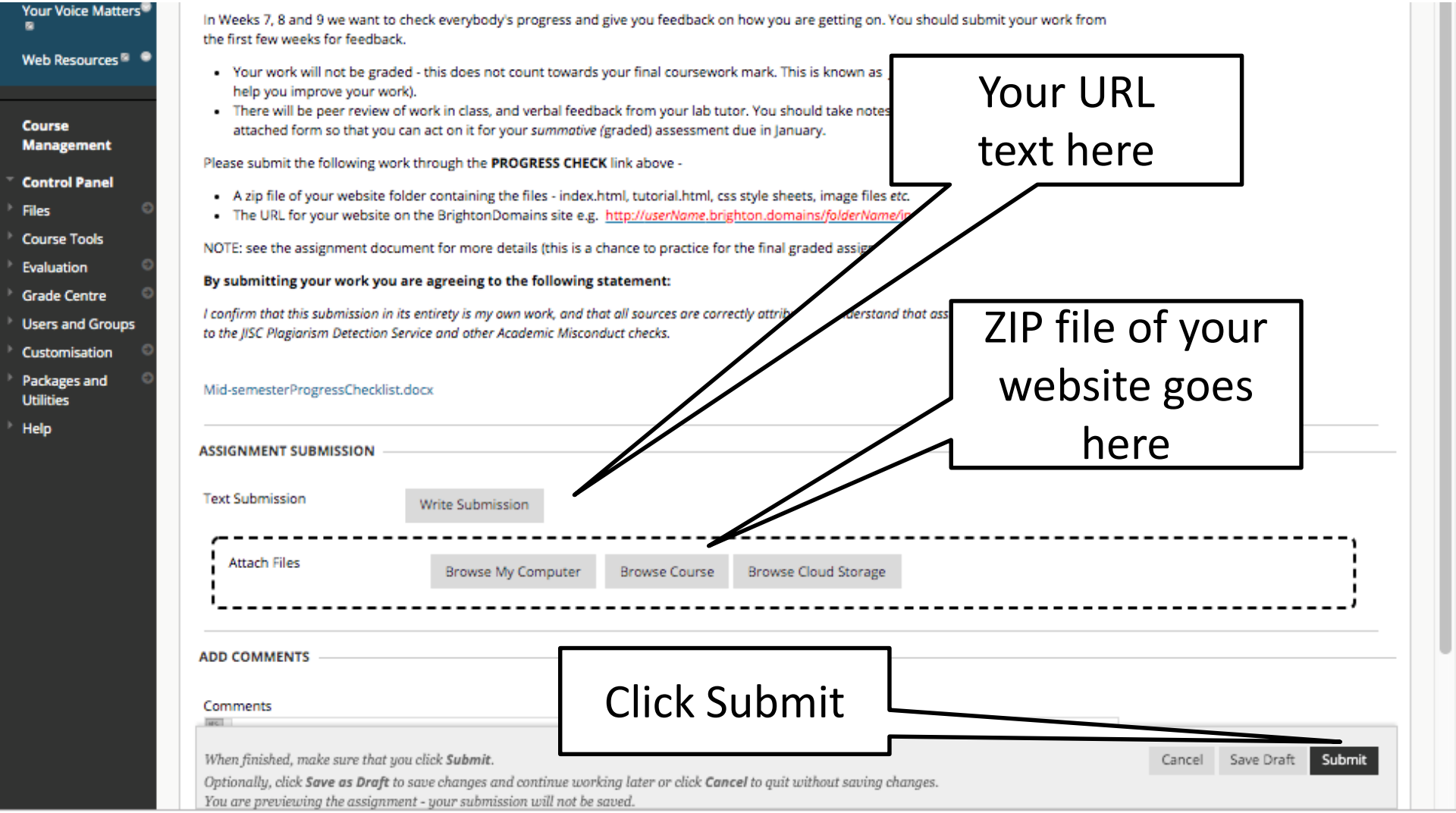
Mid-semester Progress Check

Click this link to submit work

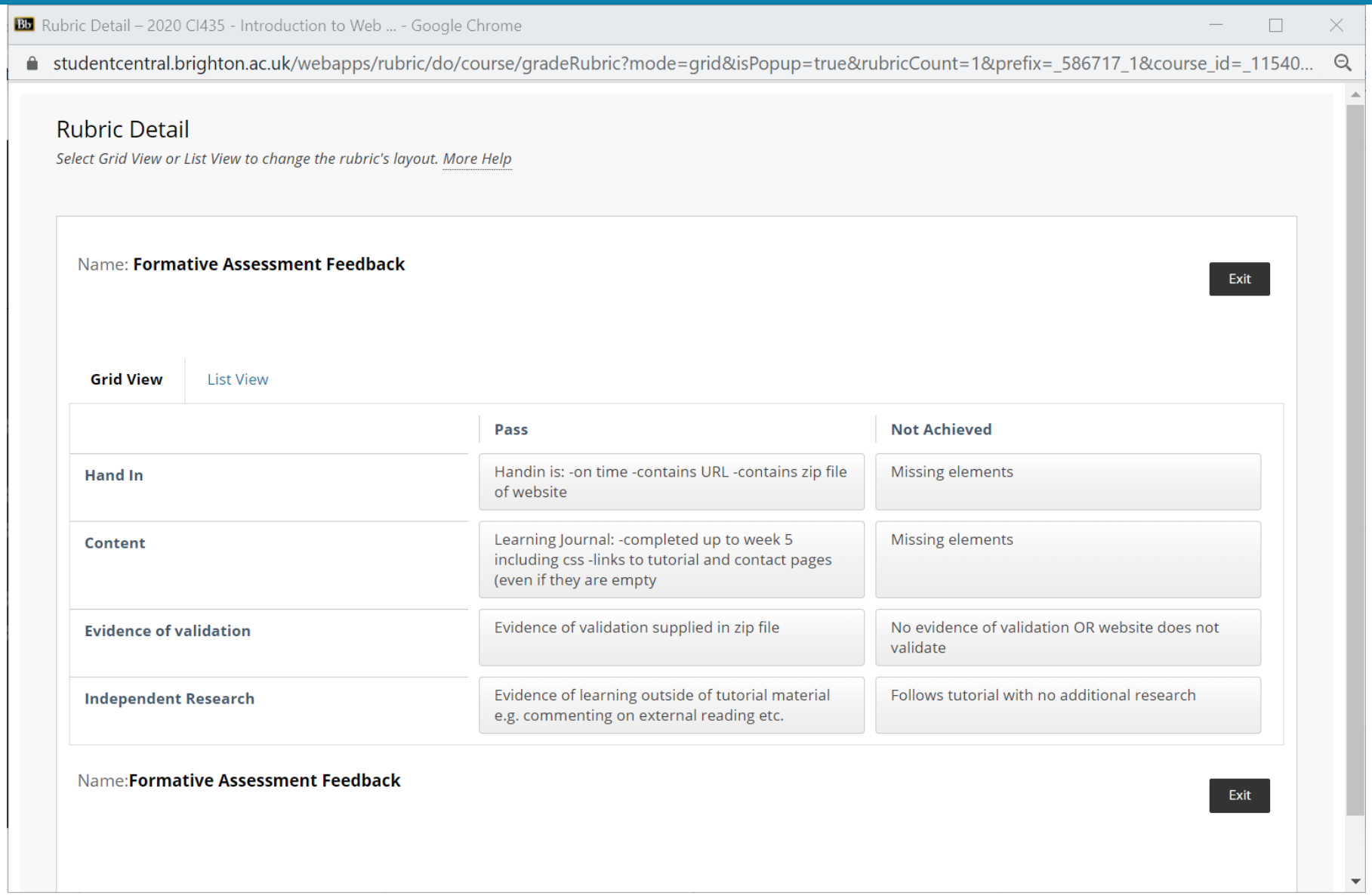


Link to the Assessment page

Mid-semester Progress Check



Mid-semester Formative Progress Rubric Checklist



Mid-semester Progress Check

YOUR JOB

- Upload your website as a zip file
- Include your URL in the text submissions section
- Attend tutorials in the next 3 weeks
- Record verbal feedback that you are given

TUTOR'S JOB

- Review your website and give basic feedback via rubric
- Provide verbal feedback

The Tutorial web page

- Write an **original** tutorial – a study skill of your choice
- **Does the *Tutorial* page have to be styled differently from the *Learning Journal*?**
  - The *Tutorial* doesn't need a separate style sheet
  - Content and structure will differ – the *Tutorial* is written for a different audience and has different content – so there should be some CSS that *only* applies to this page
  - You might want to lay it out differently
- I will be covering some of the features you might include – *e.g.* embedded video, a responsive image gallery - in the next weeks

This lecture will cover…

- Introduction to responsive web design/development (RWD)
- RWD – 3 core techniques
- The `viewport meta` element
- CSS3 media queries
  - Syntax
  - Breakpoints
  - Testing
- Responsive *Learning Journal* – CSS grid layout workflow
- Reading and web resources

Mobile web access

Do you have a smartphone?

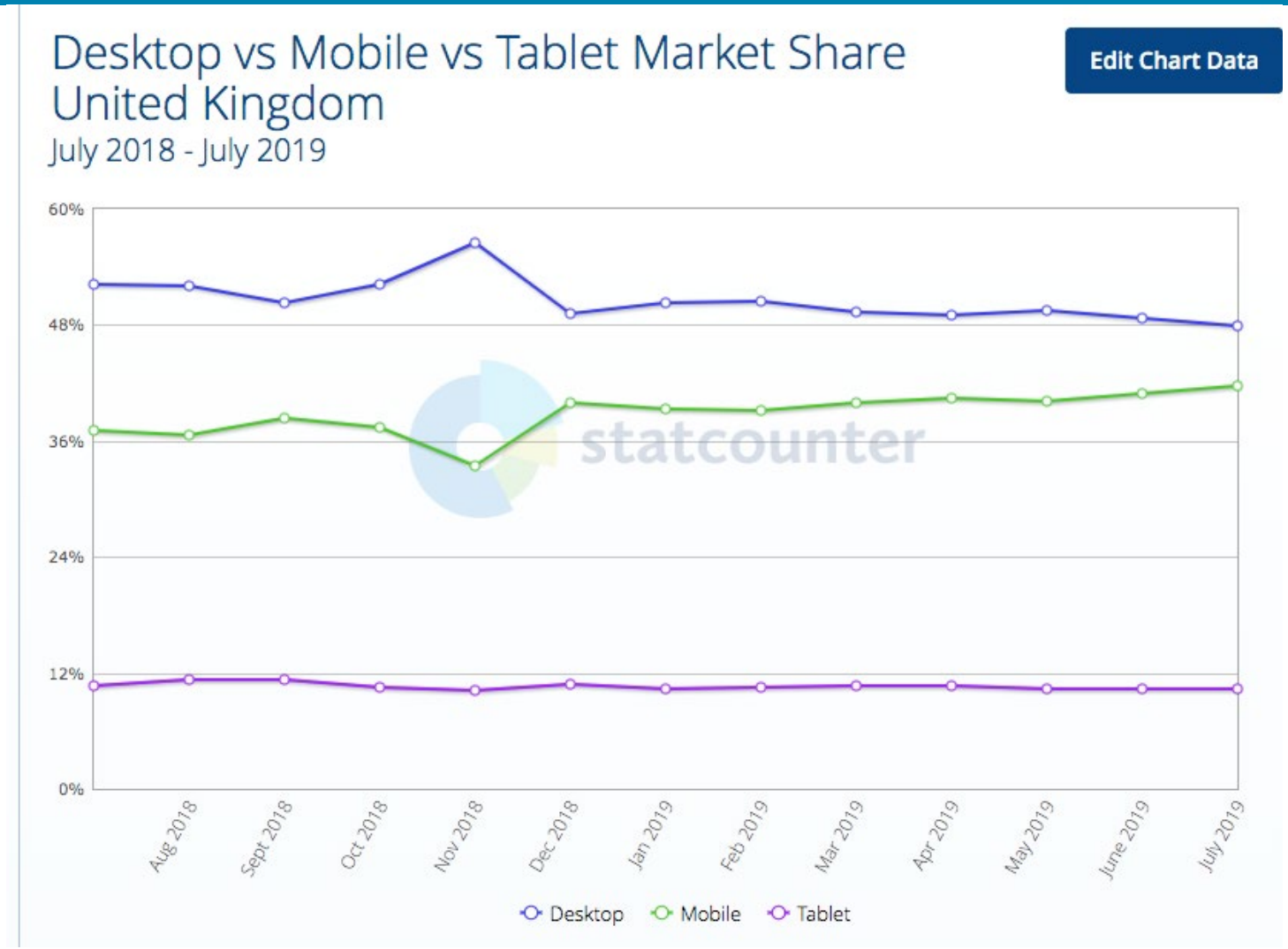
How often do you access the web on your mobile?

Where are you when you access the web on your mobile?

What are your goals?

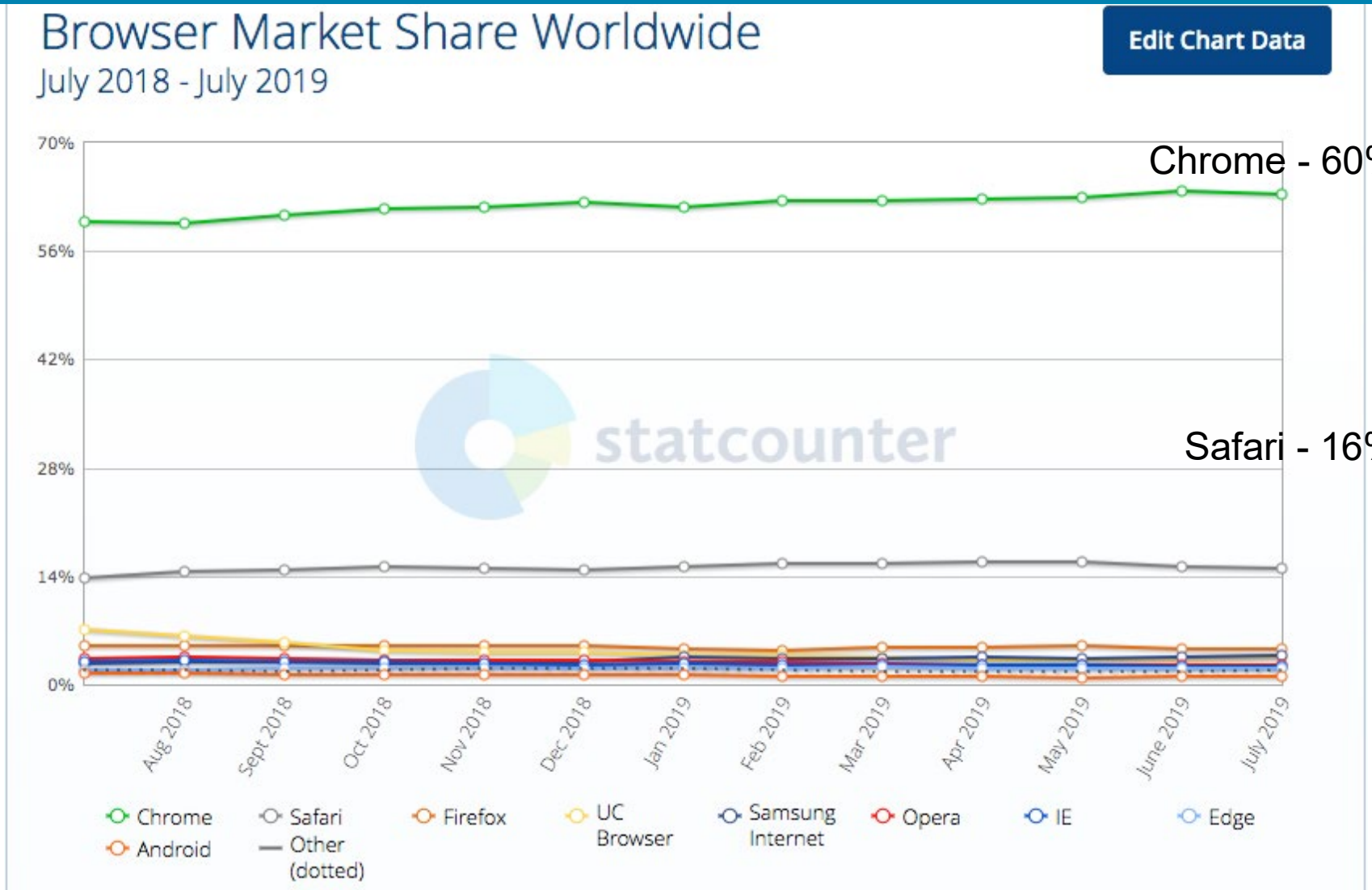
Is this different from when you access the web on a desktop or laptop?

UK - platform market share



https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/united-kingdom

UK - browser market share



https://gs.statcounter.com/browser-market-share

Responsive web design

- The rapid growth in mobile devices to access the web has driven a **"paradigm change"** in web design and development
- For any web project ask –
  - Will site be accessed by a small screen mobile device?
  - Will users access the site by more than one type of device?
  - Will users expect the same *quality* of user experience and content from mobile, tablet and desktop web access?
- Web designers and developers have no control over how users are accessing the web – it's important to design a user experience that caters for all

Responsive web design

- Not affordable for most businesses to create duplicate websites for desktop and mobile
- Large number of variables – large screen smart TVs, e-book reader, games consoles, desktop, laptop, hi-res retina screens, tablet, mobile – different aspect ratios, landscape/portrait screen orientation
- The *ideal* solution is one site, accessible by a range of devices, that adapts to the devices accessing it
- Usual approach today – design and build a **responsive website starting mobile first**

Responsive web design

"Rather than tailoring disconnected designs to each of an ever-increasing number of web devices, we can treat them as facets of the same experience. We can design for an optimal viewing experience, but embed standards-based technologies into our designs to make them not only more flexible, but more adaptive to the media that renders them. In short, we need to practice **responsive web design**.

But how?"

Ethan Marcotte, inventor of RWD, May 25, **2010**

http://www.alistapart.com/articles/responsive-web-design/

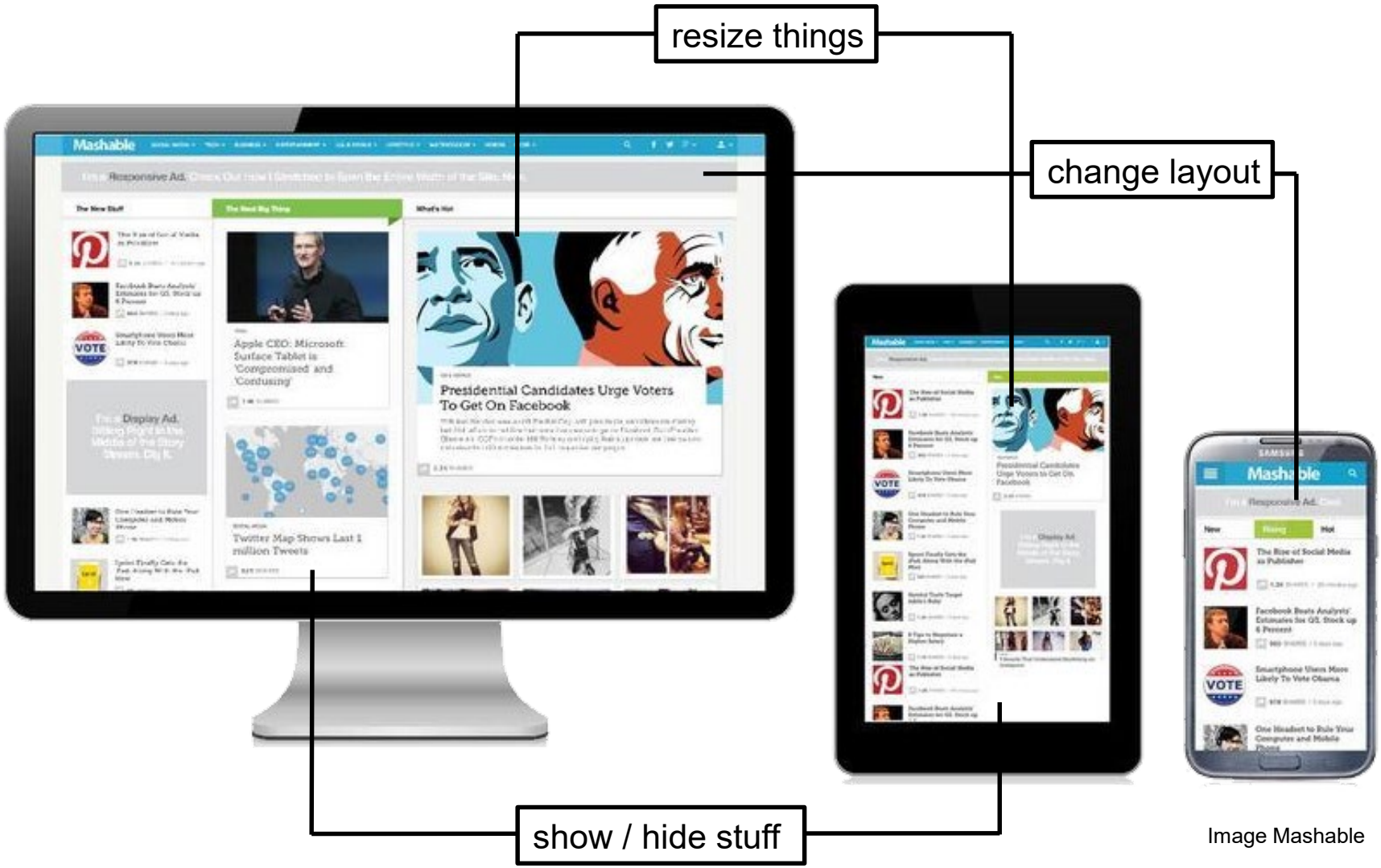


Image Mashable

E.g. This week's *Learning Journal* lab tutorial example - http://jh1033.brighton.domains/ci435/tutorials/learningJournal/indexrwd1.html

RWD – 3 techniques to learn

There are **3 core RWD techniques**

1. **Fluid layout**
   - Use these units to size elements – %, em, rem
   - CSS layout modules – grids, flexbox, multi-column
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
   - W3C Requirements for Standardizing Responsive Images (professional level detail)

Step 1: The `viewport` meta element

- **See lecture 3**
- On desktop/laptop browsers the **viewport** = width of the browser window (minus chrome, scrollbars, menu *etc.)*
  - If the viewport/browser window is made smaller the web page remains the same size; scroll bars are applied
- On mobile devices web pages are scaled down to fit the viewport/screen width and users have to 'pinch and zoom'
  - Very poor user experience
- First step is to make a mobile browser behave like a desktop browser

Step 1: The `viewport` meta element

- The `viewport` attribute for the `meta` element allows the mobile device viewport to be set to a specific width
- Write this line in the `<head>` of your HTML documents before the `<link>` to the style sheet(s) -

```
<meta name="viewport" content="width=device-width, initial-scale=1"/>
```

- `"width=device-width"` - sets the pixel width of the viewport to be equal to the screen width of the device – so that a mobile browser will behave the same as a desktop one
- `"initial-scale=1"` – tells the browser to render the web page at full-size when it's loaded, rather than scaling it down

Step 2: CSS3 media queries

- CSS3 **media queries** are what make the web page **responsive**
- Write CSS style rules that are tailored to the capabilities of the different devices being used to view the page
- *E.g.* - in my *Learning Journal* example a media query targets wide screen devices with CSS for a horizontal menu

```
@media (min-width: 50em;)
 {
   .menu li {
   display: inline;}
 }
```

Query targets viewport with a width **equal to or greater than 50em** (800px)

CSS rule to make a horizontal navigation

CSS3 media queries

- Media queries inspect the **capability** of the device that is being used to access a web page, checking for features like -
  - width and height of the **viewport**
  - **device** screen width and height
  - **orientation** of the screen – in landscape or portrait mode
  - screen **resolution**
  - **aspect ratio** (proportional relationship between display width and height)

CSS3 media queries

- A media query consists of a **media type** and one or more **expressions** that check for particular media features such as width, height, screen resolution
- A media query is a **logical expression** that is true or false
- A media query is true…
  - *if* the **media type** of the media query matches the media type of the device where the user agent is running
  - *and* all the **expressions** in the media query are true
- When a media query is true the browser applies the CSS styles that correspond to that query
- For examples of how to write media queries and their syntax - https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries

CSS3 media queries

E.g. simple media query to target viewport width -

```
@media screen and (min-width: 320px) {
    body { color: #000;}
}
```

- **`@media screen`** - checks whether the **media type** of the device accessing the web page **`=screen`**
- The **logical keyword** **`and`** ensures that both the media type *and* the conditions in the expression must **= true**
- The expression is in brackets; the **condition** **`min-width`** tests the width of the viewport – to find out if width is equal to, or greater than 320px
- **CSS style rules** are nested within the query, in curly braces, and are applied if the query is 'true'

CSS3 media queries

- Media queries can be written in HTML, in the document head, to load an external style sheet that only applies to specific devices

  ```
  <link href="print.css" rel="stylesheet" media="print"/>
  ```

  i.e. the style sheet 'print.css' is used if the **media type** of the device accessing the web document = **printer**
- It's now more usual to use the **`@media`** rule in the style sheet to target different media types
  - This avoid having to request an additional file when the web page loads - improves performance of the website

CSS3 media queries

- Use the **`@media`** rule to target different media types, and screen sizes, from a single style sheet
- To target a printer add this media query and CSS print styles to the end of your style sheet -

```
@media print  {
 /* All print styles go here */
   #header, #footer, #nav { display: none;}
}
```

- The CSS rules for a printer only apply if the `@media` query is 'true' - *i.e.* the device accessing the web page is a printer

CSS3 media queries: `min-` and `max-`

- **`min-`** equal to or greater than
- **`max-`** equal to or smaller than
- *E.g.* to target a mobile screen –

```
@media screen and (max-device-width: 480px){}
```

- This line combines the media type 'screen', with a query (in parentheses) enquiring if the horizontal width of the screen – '`max-device-width`' - is **equal to or less than 480px**
- *If* the media type is screen *and if* the result of the query is **TRUE** then the mobile style sheet will be applied by the browser

CSS3 media queries: breakpoints

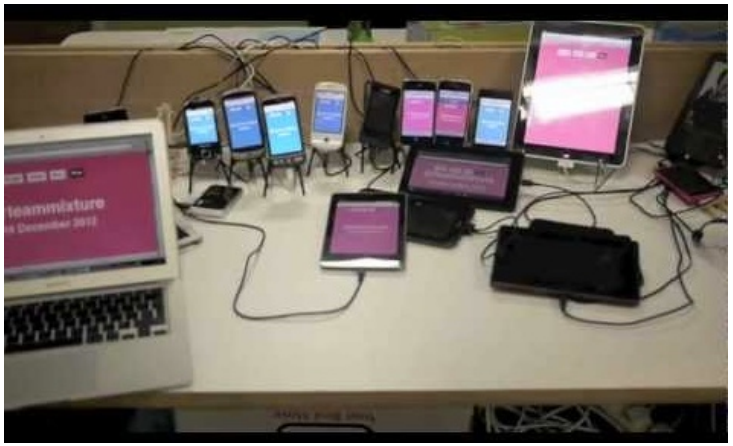
- Typical, basic media queries test for the viewport `max-width` and/or `min-width` and apply a series of rules to optimise presentation
- Standard widths are often used as **breakpoints** – 320px (iPhone 5, 6), 768px (iPad), 1224px (desktop, laptop); 480px – iPhone 5, 6 in horizontal orientation
- **Future-proof sites by creating breakpoints that work with content, rather than specific devices – *e.g.* 300px, 600px**
- If you are using em or rem as your unit of measurement pixel values can be converted
- The best process is to test breakpoints in the browser and test them until you get the desired result

CSS3 media queries: testing

- Once your web page is responsive test it on a smartphone mobile browser, as well as a desktop and tablet
- Developer tools for testing built into the browser -
  - **Chrome Dev Tools** - Device Mode feature https://developers.google.com/web/tools/chrome-devtools/device-mode/
  - **Firefox developer tools** - Responsive Design Mode https://developer.mozilla.org/en-US/docs/Tools/Responsive_Design_Mode
- Try these tools in your lab class this week

CSS3 media queries: testing

- Online smartphone and tablet emulators -
  - **ProtoFluid** - an app for testing responsive prototype web pages by URL and viewing them as if on a range of devices – iPad, iPhone *etc.*
  - http://protofluid.com/
  - http://mobiletest.me/
- Mobile friendly test - https://www.google.co.uk/webmasters/tools/mobile-friendly/
- Mobile speed test - https://testmysite.withgoogle.com/intl/en-gb/
- Essential to test on as many **actual devices** as possible



CSS - responsive *Learning Journal* page

- This week's 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 is a collection 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 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

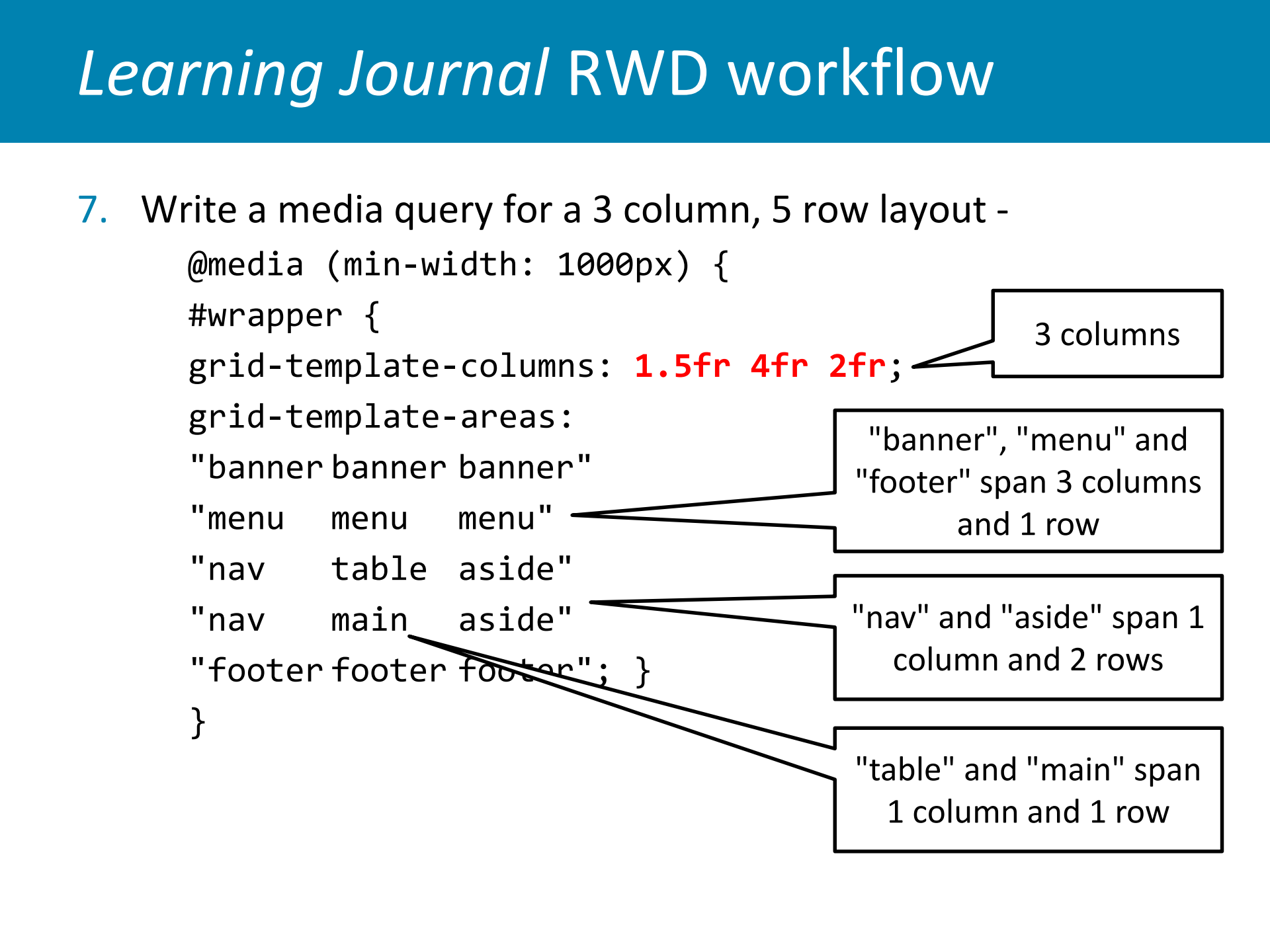
"nav", "table", "aside" and "main" span 1 column and 1 row

- 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 not display 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

This week's reading

- Marcotte, E., 2010. Responsive web design. *A List Apart.*

  http://alistapart.com/article/responsive-web-design
- **MDN**: Media Queries –

  https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries
- **MDN**: CSS grid layout -

  https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Grids
- Responsive *Learning Journal* Example -

  http://jh1033.brighton.domains/ci435/tutorials/learningJournal/indexrwd1.html
- Linked In Learning - CI435 playlist - responsive web design courses