Using CSS to build layouts

RENDERING PAGES

* Sometime we really want to have more control over our layout
* We want to place our HTML elements in particular locations on screen
* This creates tension: Browsers are built around layout engines which are optimised for presenting documents
  + Particularly documents that have a quasi-academic publication structure (intro, sections, conclusion)
  + Text is inherently responsive - it can keep wrapping to the next line until you reach the end - resizes nicely and doesn’t care about viewport/window/screen/device size
  + When we try to use this engine for more exacting layouts - trying to implement desktop style interfaces - difficulties arise

HOW NOT TO LAY OUT PAGES

* So many ways that HTML has been misused to try to get pixel-perfect layouts in quest for desktop style UI
  + e.g. HTML table + single pixel transparent gif files
    - Use as many transparent gifs to push elements around the page until they are all in place regardless of window size
    - This is a horrible solution to the problem of getting HTML layout engines to do something they weren’t designed to.
  + e.g. <divs> to group elements then CSS to arrange them
    - Better
    - Still horrible - breaks semantic HTML

NEW CSS SOLUTIONS

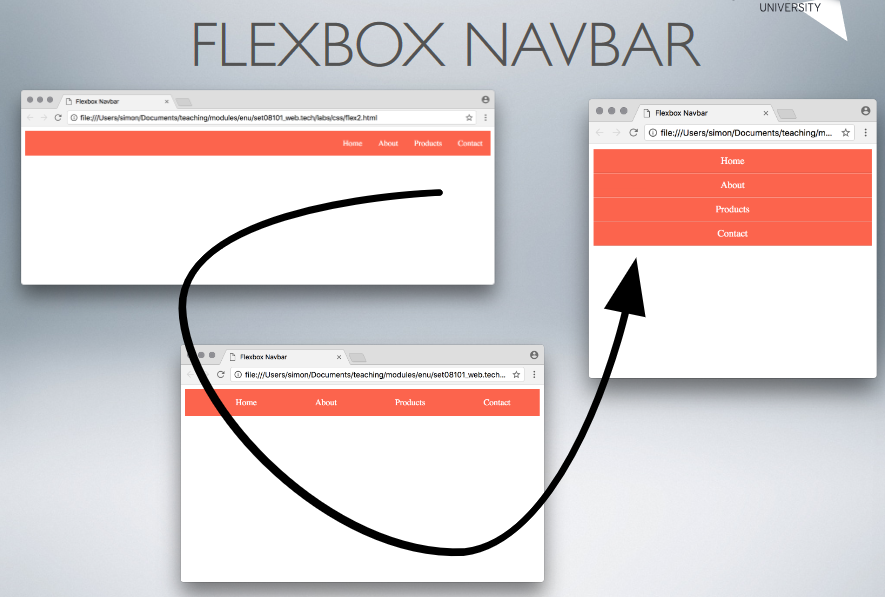
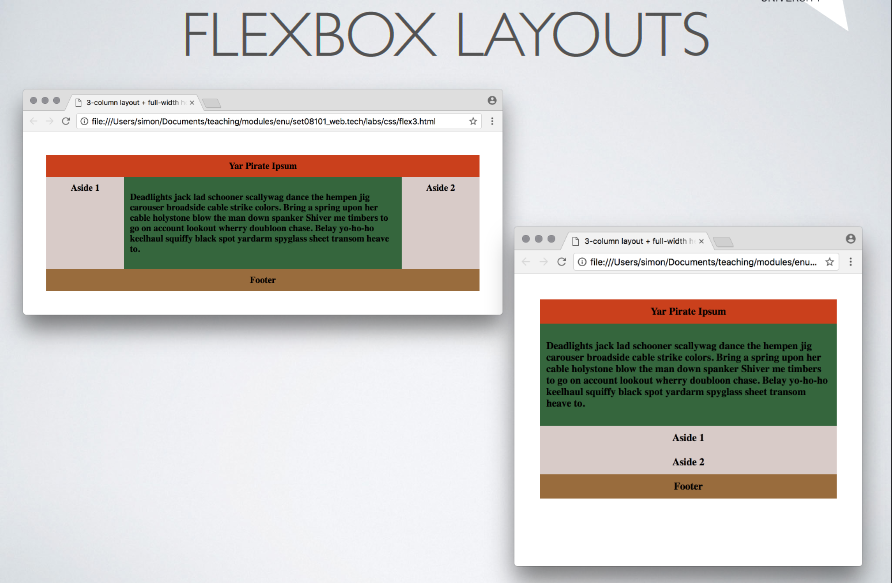
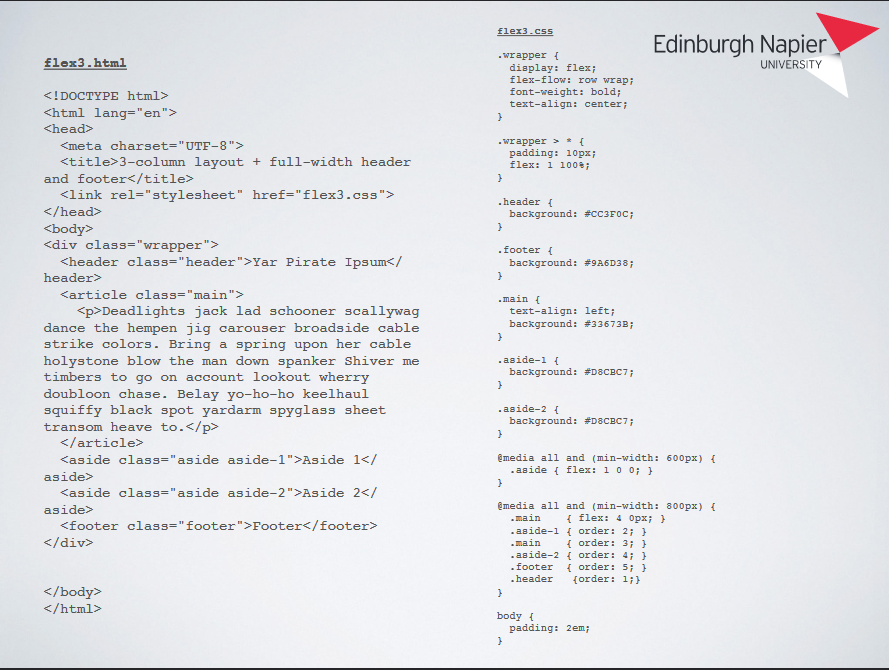
* Two recent solutions give better control over relative placement of HTML elements...
* ...whilst maintaining much of the basic flexibility and responsiveness of traditional HTML
  + Flexbox - Way to lay out linear collections of elements either horizontally or vertically
  + Grid Layout - Way to lay out matrices of elements as rows and columns

FLEXBOX & GRID LAYOUT

* Designed to use CSS to influence the placement of elements
* Take into account:
  + How we build interfaces - certain recurrent patterns of user interface layout
  + How pages are rendered - from upper left corner of screen to lower right corner of screen
  + Screen has two dimensions: horizontal and vertical
  + Can lay out things in 1 dimension, either horizontally or vertically (but not both) – leads to flexbox - a single dimensional collection
  + Can lay out things in 2 dimensions, both horizontally & vertically - a dimensional collection

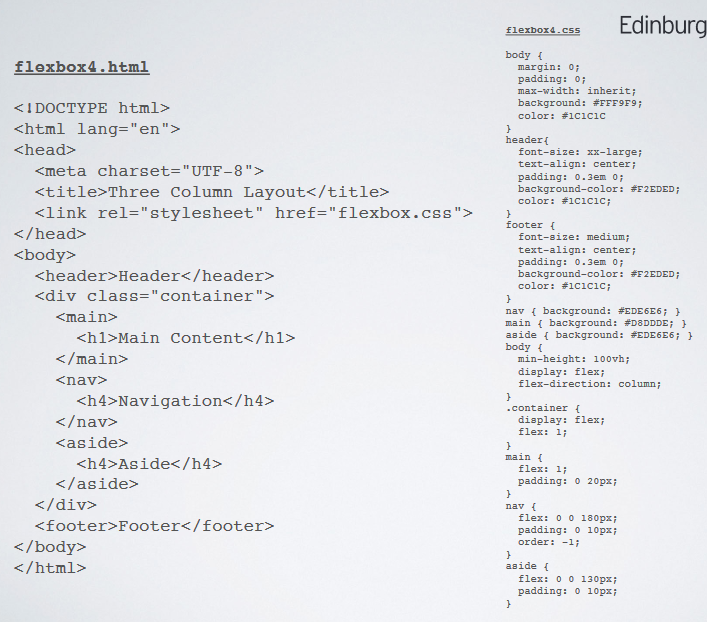
FLEXBOX

* Imagine a box reaching horizontally from one side of the screen to the other
* This box can contain HTML elements
* We want to control how elements are spread across the box from one side of the screen to the other - e.g. control spacing to either side of, and between, elements
* If screen is too narrow - want to control how the box flows over into another row and arranges elements below
* Flexbox let’s us do this
* Diagram

  Description automatically generated
* 
* 
* 
* 
* 

FLEXBOX UI

* With a nicer colour palette & a little differentiation in text sizes in the various sections...
* ... We can build some pleasing & elaborate user interfaces:
  + Graphical user interface, application

    Description automatically generated
  + 

FLEXBOX PROPERTIES OVERVIEW

* flex
* flex-basis
* flex-direction
* flex-flow
* flex-grow
* flex-shrink
* flex-wrap

GRID LAYOUT

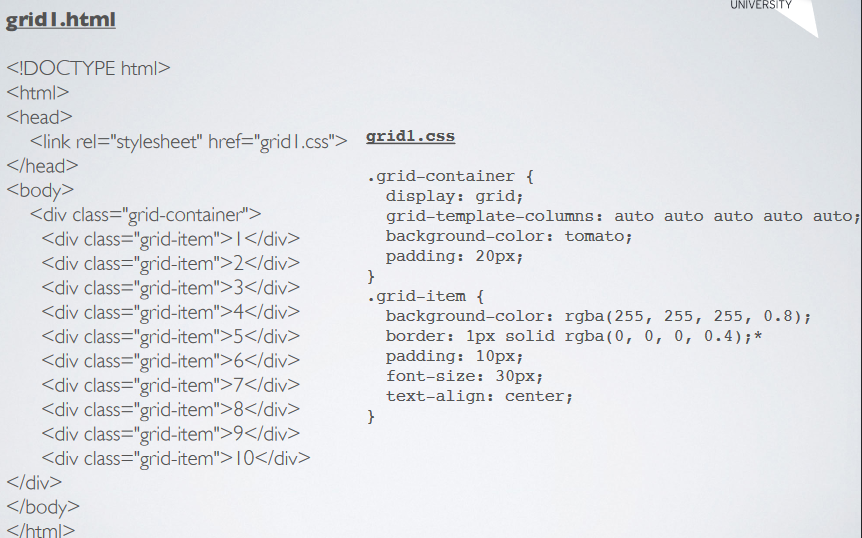
* Similar in some ways to the HTML table
* HTML table has a semantic meaning - data within table is related in various ways and organised according to rows & columns
* Grid layout has a presentational meaning but the members of the grid have no semantic relation beyond their placement relative to each other
* e.g. grid elements in the same row or column aren’t related

USING GRID

* We use Grid by setting the display: grid; property on a containing element, e.g. a <div> or even the <body>
* Grid will then be used to layout the child elements within the containing element
* Grid has a number of related properties & values that can be used to control the layout of the grid,
  + e.g. spacing between elements, flow of elements, arrangement of elements

REGULAR LAYOUTS

* Behaviour of grid is different on resizing compared to flexbox:
  + Graphical user interface, application

    Description automatically generated
  + 

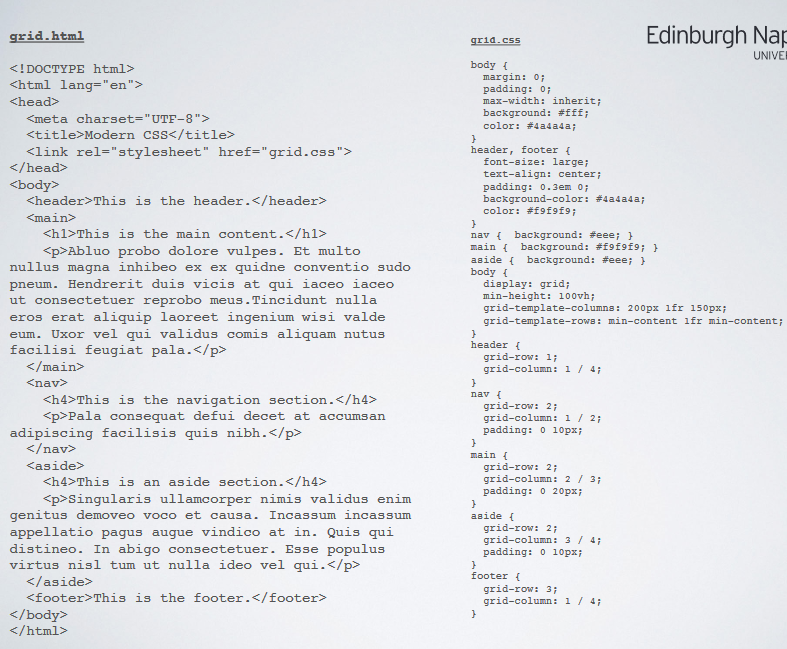
IRREGULAR LAYOUTS

* Can enable individual elements to span multiple rows or columns, e.g. using grid-column-start & grid- column-end
  + Table

    Description automatically generated with medium confidence
  + 

GRID LAYOUT FOR UI

* Similar to our flexbox based layout
* Doesn’t reflow in the same way as for flexbox
* More control over positioning of elements relative to others within the grid
* (at the expense of flexibility & responsiveness)
* The more we try to override default HTML layout engine in the browser the more we have to return to previous responsiveness
* Graphical user interface, text, application, email

  Description automatically generated
* 

GRID PROPERTIES OVERVIEW

* grid
* grid-area
* grid-auto-columns
* grid-auto-flow
* grid-auto-rows
* grid-column
* grid-column-end
* grid-column-gap
* grid-column-start
* grid-gar
* grid-row
* grid-row-end
* grid-row-gap
* grid-row-start
* grid-template
* grid-template-areas
* grid-template-columns
* grid-template-rows