CS1115/CS5002

Web Development 1

Dr Derek Bridge

School of Computer Science & Information Technology
University College Cork

Types of layout

- **Fixed-width** or **liquid**
  - Set the width in pixels for fixed-width or percentages/ems for liquid
- Single column or multi-column
  - HTML tables, CSS floating, CSS positioning (absolute, relative, fixed), CSS Flexbox layout or CSS Grid Layout

Fixed-width

- Rigid: they don't change size
- Pro: consistent look
- Con: some visitors will see acres of space or horizontal scrollbars
- Use:
  - when you know width of majority of your visitors' windows
  - your site has lots of fixed-width content, e.g. images
  - pixel-perfect appearance is paramount to you

```
html {
    background-color: white;
}

body {
    width: 960px;
    margin: 0 auto;
    background-color: green;
}
```

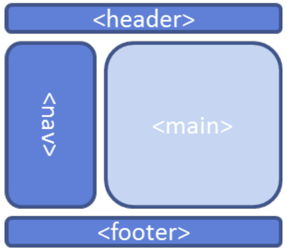
Liquid

- Adapt to user's viewport
- Pro: respect user's preferences and tries to make best use of the space
- Con: hardest to get right, esp. when there are images, etc.
- Use: most of the time

```
html {
    background-color: white;
}

body {
    width: 80%;
    margin: 0 auto;
    background-color: green;
}
```

Running example of two-column layout

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf8" />
        <title>A simple document</title>
        <link rel="stylesheet" href="styles.css" />
    </head>
    <body>
        <header>
            Header
        </header>
        <nav>
            Nav
        </nav>
        <main>
            Main
        </main>
        <footer>
            Footer
        </footer>
    </body>
</html>
```



HTML tables for multi-column layouts

- Ancient web sites and web sites produced by novices use HTML tables for multi-column layouts
- **Don't!**
- **Question** Why not?

```
<table>
    <tr><td colspan="2">Header</td></tr>
    <tr><td>Nav</td><td>Main</td></tr>
    <tr><td colspan="2">Footer</td></tr>
</table>
```

CSS floating

- The main idea is to set an image to the left or right, with the text wrapping around the image

```
<p>
    Wombats are short-legged, muscular quadrupedal marsupials that are
    native to Australia. <img src="wombat.jpg" alt="A cute wombat" />
    They are about 1 m (40 in) in length with small, stubby tails.
    …
</p>
```

```
img {
    float: left;
}
```

Wombats are short-legged, muscular quadrupedal marsupials that are native to Australia. They are about 1 m (40 in) in length with small, stubby tails. There are three extant species and they are all members of the family Vombatidae. They are adaptable and habitat tolerant, and are found in forested, mountainous, and heathland areas of south-eastern Australia, including Tasmania, as well as an isolated patch of about 300 ha (740 acres) in Epping Forest National Park in central Queensland. (From Wikipedia)

CSS floating for layout

- Two-column layout at its simplest, e.g.

```
nav {
    float: left;
    width: 25%;
}

main {
    margin-left: 25%;
}

footer {
    clear: both;
}
```

- But there are lots of issues!
- E.g. in more complex cases, you may also need to stop the floated element from overflowing its container, either with a clearfix hack or by making the container a block formatting context using `display: flow-root`

CSS positioning

- These properties allow you to precisely position an element
- The box offset properties (`top`, `right`, `bottom` and `left`) specify *where* an element will be positioned relative to some container
- The `position` property specifies *how* an element will be positioned, roughly:
  - `static` (the default): normal position in the flow (the box offset properties have no effect)
  - `relative`: the element is 'nudged' from its normal position in the flow, with space left behind
  - `absolute`: the element is removed from the flow and positioned relative to its nearest ancestor that is not `static`, if there is one, else the viewport (e.g. browser window)
  - `fixed`: like `absolute` but always relative to the viewport
  - `sticky`: the element is positioned `relative` until a given offset position is met in the viewport; then it sticks, like `fixed`

CSS positioning for layout

- An example of two-column layout:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf8" />
        <title>A simple document</title>
        <link rel="stylesheet" href="styles.css" />
    </head>
    <body>
        <header>
            Header
        </header>
        <div id="container">
            <nav>
                Nav
            </nav>
            <main>
                Main
            </main>
        </div>
        <footer>
            Footer
        </footer>
    </body>
</html>
```

```
#container {
    position: relative;
}

nav {
    position: absolute;
    top: 0;
    left: 0;
    width: 25%;
}

main {
    margin-left: 25%;
}
```

- This 'corrupts' the HTML (an extra `<div>`) just to get the CSS to work
- There remains a problem with the `<footer>`
- Because of these problems, few people use positioning for page layout; if they use it at all, they use it to position small design elements such as images in non-standard places

CSS Flexbox

- A flexbox contains items, which are laid out in a one-dimensional way
  - You choose whether to lay them out in a row (horizontally) or in a column (vertically)
  - You choose what to do when there is not enough space (e.g. wrap) or too much space
- Properties for the flexbox:
  - `display: flex`
  - `flex-direction`: e.g. `row` (default), `column`, …
  - `flex-wrap`: e.g. `nowrap` (default), `wrap`, …
  - Others: `justify-content`, `align-items`, `align-content`
- And properties for the items in the flexbox, e.g. `order`, `flex-grow`, `flex-shrink`, `flex-basis`, `align-self`
- Tutorials: A Complete Guide to Flexbox; Understanding Flexbox
- Game: Flexbox Froggy

Flexbox for an image gallery

```
<section>
    <img src="image001.jpg" alt="…" />
    <img src="image002.jpg" alt="…" />
    <img src="image003.jpg" alt="…" />
    <img src="image004.jpg" alt="…" />
    …
</section>
```

```
section {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: flex-start;
}
```

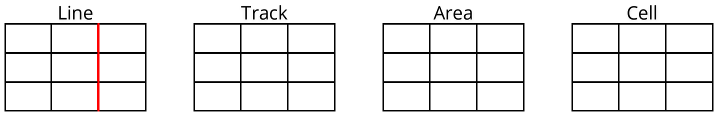
Flexbox for a two-column layout

```
body {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
}

header, footer {
    flex-basis: 100%;
}

nav {
    flex-grow: 1;
}

main {
    flex-grow: 3;
}
```

However, Flexbox layout is most suited to simple layouts — use Grid layout for more complex layouts

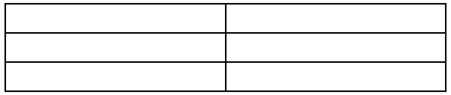
CSS Grid Layout

- Grid Layout makes two-dimensional layouts easy (easier)
- It is quite new
- A grid contains items arranged in columns and rows



- Tutorials: A Complete Guide to Grid; Grid By Example
- Game: Grid Garden
- In the future, CSS will also support subgrids

Grid layout for a two-column layout



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

header {
    grid-area: top;
}

nav {
    grid-area: middle-left;
}

main {
    grid-area: middle-right;
}

footer {
    grid-area: bottom;
}
```