CS1115/CS5002

Web Development 1

Dr Derek Bridge

School of Computer Science & Information Technology
University College Cork

Boxes

- Every HTML element generates zero, one or more CSS **boxes**
  - Visually, the tree of elements becomes a tree of boxes
  - Mostly, one box per element but a few elements have more than one box and some have none
  - E.g. the `<li>` element has two. **Q**: Why?
  - When displayed, a box might be broken across lines
- In CSS, we can style the boxes

Display type

- CSS's `display` property defines a box's **display type**
- The two main display types:
  - `block`:
    - the box starts on a new line
    - it takes up the full width available (i.e. the width of its container)

    — hence they stack vertically
  - `inline`:
    - the box continues on the current line
    - it takes up only as much width as necessary for its content

Display types

- The `display` property has over 20 possible values, in addition to `block` and `inline`
- The CSS initial value for all boxes is `inline`
- But your browser's default stylesheet applies a more sensible value, e.g.:

```
header, nav, main, section, article, aside, footer, figure, figcaption, div, p,
h1, h2, h3, h4, h5, h6, ul, ol, form, fieldset {
    display: block;
}
```

  leaving, e.g., `em, i, strong, b, span, a, img,`… as inline
  (As mentioned previously, `<img>` being inline surprises many people)
- Of course, you can change them in your own stylesheet, e.g.

```
img {
    display: block;
}
```

  Question: What did we do in earlier lecture to make images appear on their own lines?

The Box Model

Margin — space outside the border

Border — a 'frame' between padding and margin

Padding — space between content and border

Content — where text and images appear

Example

```
<p>
    To bring the spirit of the Caribbean to
    your party, why not serve Mojitos? Sweet
    but cool, they're sure to get you in
    touch with your Latin side.
<p>
```

```
p {
    margin: 1em;
    border: 0.0625em solid black;
    padding: 2em;
    width: 30em;
}
```

The content

- The `width` of a box refers to the width of the content, not the whole box!

```
p {
    width: 30em;
}
```

- The *total width* of the box = left margin + left border + left padding + content + right padding + right border + right margin
- What units can you use?
  - `em`, `rem`, `px`, ...
- You can also use percentages (e.g. `width: 80%`)
  - ... in which case, the width is based on the width of the element's container
- And the same with `height` — although you should be wary of setting heights
- (Advanced: setting the width or height of a non-replaced inline element has no effect)

Padding

- Four properties whose values can be in ems, pixels (px) or other units

```
p {
    padding-top: 0.5em;
    padding-right: 1em;
    padding-bottom: 0.5em;
    padding-left: 1em;
}
```

- Various shorthand versions, including:

```
p {
    padding: 0.5em 1em 0.5em 1em;
}
```

```
p {
    padding: 0.5em 1em;
}
```

```
p {
    padding: 0.5em;
}
```

- If you use percentages, they are based on the *width* of the element's container

Background colours and images

- The `background-color` property affects the content + padding only, e.g.

```
p {
    background-color: blue;
}
```

- The `background-image` requires a URL:

```
p {
    background-image: url("images/photo.jpg");
}
```

- By default, the image is repeated so that it covers the content + padding
- But you can control the repetition by setting values for `background-repeat`, `background-attachment` and `background-position` or you can scale the image using the `background-size` property
- There is a shorthand using a property called `background`
- Background colours display behind background images, which display behind content

Rounded corners on borders

- For rounded corners, specify the `border-radius`:

```
p {
    border: 0.0625em solid blue;
    border-radius: 1em;
}
```

- If you give two numbers, the corners are ellipses rather than circles:

```
p {
    border: 0.0625em solid blue;
    border-radius: 2em 1em;
}
```

  - the first value specifies the horizontal radius
  - the second value specifies the vertical radius
- You can style individual corners using, e.g., `border-top-right-radius`
- Similarly for top-left, bottom-right and bottom-left

Borders

- You must specify colour, width and style (e.g. solid, dashed, double,...):

```
p {
    border-width: 0.0625em;
    border-style: solid;
    border-color: black;
}
```

- Shorthand version:

```
p {
    border: 0.0625em solid black;
}
```

- **If you don't set the border style, you will see nothing!**
- You can set individual parts of the border using, e.g., `border-top-color`, `border-top-width`, `border-top-style`
- Similarly for `border-right`, `border-bottom` and `border-left`
- If interested, look up the `border-image` properties

Box shadows

- To obtain a simple shadow:

```
p {
    box-shadow: 2em 3em gray;
}
```

- The three values are:
  1. The horizontal offset of the shadow, where positive means the shadow will be on the right of the box, and negative will put the shadow on the left of the box
  2. The vertical offset of the shadow, where positive means the shadow will be below the box, and negative means above
  3. Colour
- An 'inner' shadow is possible by incuding the word `inset`
- Two other numbers (a blur radius and spread radius) may also be specified
- Shadows do not affect the total size of the box

Margin

- Margin is done similarly to padding, with the same shorthand versions

```
p {
    margin: 1em 2em 1em 2em;
}
```

- Again, if you use percentages, they are based on the *width* of the element's container
- Watch out! Vertically adjacent margins are collapsed
    - Suppose the bottom margin of a box is 5px
    - Suppose the top margin of the box below is 10px
    - The amount of space will **not** be 15px
    - It will be 10px: the smaller margin is eliminated