COMSM0085

Overview of Software Tools

Software Tools: Part 2

(COMS10012 / COMSM0085)

Last week (Week 6): HTTP & HTML

This week (Week 7): CSS

Next week (Week 8): JS

CSS

Cascading Style Sheets (we'll get to the 'cascading').

A set of rules that define how HTML elements are displayed.

In most cases, rules refer to the setting of visual properties for a HTML document element (more technically: to the DOM objects derived from the HTML).

CSS composition

CSS entries follow fairly simple syntax:

```
selector {
    property : value;
}
```

- `property` is any display property that is meaningful for this element type.
- `value` permitted values will depend on the property.
- `selector` defines which elements this rule can be applied to.

CSS implementation

Your CSS rules should be placed in a stylesheet document e.g., `mystyle.css` .

The browser can then be informed to use a particular stylesheet by a reference in the HTML document.

```
<html>
 <head>
    <link rel="stylesheet" href="mystyle.css"/>
 </head>
 <body>
  ...
```

You can also just instruct your browser to apply a custom stylesheet to HTML documents by default (‘user’ or ‘custom’ styles). But as a web designer you assume your website visitors want to see the webpage the way you intended it to be viewed.

CSS simple example

```
p {
 color : red;
}
```

For all `p` elements, set the `color` (font colour) to `red` .

Selectors

Wide range of selection capabilities.

Simplest selector: name of a single tag (e.g, `p` , `a` , `div`). Applies to all elements of that kind.

Next-simplest selector is a list of tags:

```
p, div, main {
    color : red;
}
```

Selectors: Class

Syntax for selecting only elements with particular `class` properties:

```
p.important {
   color: red;
}
```

```
<p class='important'>This is red</p>
<p>This is not</p>
```

Selectors: Class

Can also apply to `class` *regardless of element type*:

```
.important {
   color: red;
}
```

```
<p class='important'>This is red</p>
<p>This is not</p>
<span class='important'>But this is</span>
```

Selectors: ID

If you wanted to apply style to a particular document element:

```
p#uniquebox {
   color: red;
}
```

```
<p id='uniquebox'>This is red</p>
<p class='uniquebox'>This is not</p>
<p>This certainly is not</p>
```

Same as class – `#uniquebox` by itself would apply to?

Selectors: Attribute

Can generalise to select elements by any attribute.

```
p[name=tim] {
   color:red;
}
div[border=none] {
   color: blue;
}
```

`p[class='important']` would be the same as `p.important` .

Can also do some fancy partial matching, e.g., `img[title~='flower']` selects all images where the `title` attribute *contains* the string ‘flower’.

Selectors: Positional

```
<div class="container">
 <p>direct child</p>
 <div>
  <p>descendant</p>
 </div>
</div>
<p>para one</p>
<p>para two</p>
```

- A *descendant* is an element that is ‘inside’ another element, at any level.
- The *child* is an element that is *directly* contained inside the parent.
- An element *precedes* another if it comes at any point earlier at the same level of the document.
- An element *follows* another if it is the *very next* element at that level of the document.

Selectors: Positional

- `this that` (space): selects all elements `that` which are *descendants* of `this`.
- `this > that` : selects all elements `that` which are *direct children* of the parent `this`.
- `this ~ that` : selects all elements `that` which are *preceded by* an element `this`.
- `this + that` : selects all elements `that` which directly *follow* an element `this`.

Just to add complexity: all the rules can be combined.

```
div.important > p, h1#main, [title=nowred] ~ span {
    color: red;
}
```

Worth looking at a reference guide

Cascading?

Which rule applies?

```
<p class='important'>If you want to pass this unit then...</p>
```

```
p {
    font-size: 12pt;
}

p.important {
    color: red;
}
```

Values

Lots of different properties that can be set, which require different values – you will need to explore the MDN documentation to get to grips with all of the options.

However, some common elements relate to *colour* and *element layout*.

Color values

As well as `color`, you can set `background-color` and elements like `border-color`.

- Already seen `red`, and `blue`. Some other keywords for common basic colours.
- Also the hexadecimal format `#rrggbb` which accepts values from 00 to FF for each of R G and B.
- *Also* a function can be called `rgba(r,g,b,a)`, with values 1-255 for RGB and 0-1 for A.

`red` and `#FF0000` are identical. But `#FF0001` or `#FF1111` will still look 'red'.

Layout

When laying out elements on a page, a common issue relates to dealing with space 'around' an element (or between elements).

Each page element can be thought of as a 'box' with several layers:

- The **content** is the raw material of the element itself (e.g., the space for the text in a `<p>`, or for an image in an `<img>`).
- The **padding** is the space between the content and the border.
- The **border** is a (sometimes invisible) line 'around' the element, marking its bounds. It can have a thickness.
- The **margin** is the space required to be kept clear *outside* the border – other elements must not intrude on this space.

It's common to get confused between padding and margin (making the border visible helps).

Developer tools give a good visual demonstration of the values (see video for this week).

Layout values

Both `margin` and `padding` can be specified for individual sides, or collectively in clockwise order.

```
margin-top: 10px;
margin-right: 20px;
margin-bottom: 10px;
margin-left: 5px;

margin: 10px 20px 10px 5px;
```

Units of measurement

There are many different ways to specify measurement in CSS.

'Absolute' units try to produce a specific real size:

- `1 px` - 1 'pixel' (however that is interpreted: 1/96th inch).
- `1 pt` - 1 'point' (1/72th of an inch)
- `1 cm` - 1 centimetre (also `10 mm`)
- `1 in` - 1 inch

'Relative' units produce dimensions relative to either the viewport or some reference element of the page.

- `1 vh` - 1% of the viewport's height (also `1 vw` for width)
- `1 em` - 1 x whatever the size of the font (width of an 'm') is.
- `1 ex` - 1 x whatever the height of an x would be.
- `1 rem` - 1 x whatever the size of the font of the document's root element is.
- `1 %` - 1% of the size of the parent element's corresponding dimension.

Very easy to get muddled about units.

Design is hard

This unit is trying to teach you some fundamental understanding of CSS.

CSS can be hard to debug and understand – technical issues.

But successfully designing styles for real websites can also be hard in a non-technical sense. There are key principles (links to fundamentals of ergonomics, audience expectations, etc.) but fundamentally a lot comes down to questions of taste, style, fashion – web design is an *art*.

Some concepts you may find handy:

- grid-based page layouts (big focus in this week's lab)
- let designers create frameworks which you can apply (also in the lab)
- ~~stealing~~

  ideas from other websites

Exercises this week

1. Reading MDN documentation.
2. Applying basic CSS to a HTML document.
3. Getting very frustrated about pink lines.
4. Using an existing CSS framework.
5. Reading **even more** MDN documentation.
6. Using a grid layout.
7. Creating a *responsive* layout.