UNIVERSITY OF
Southampton

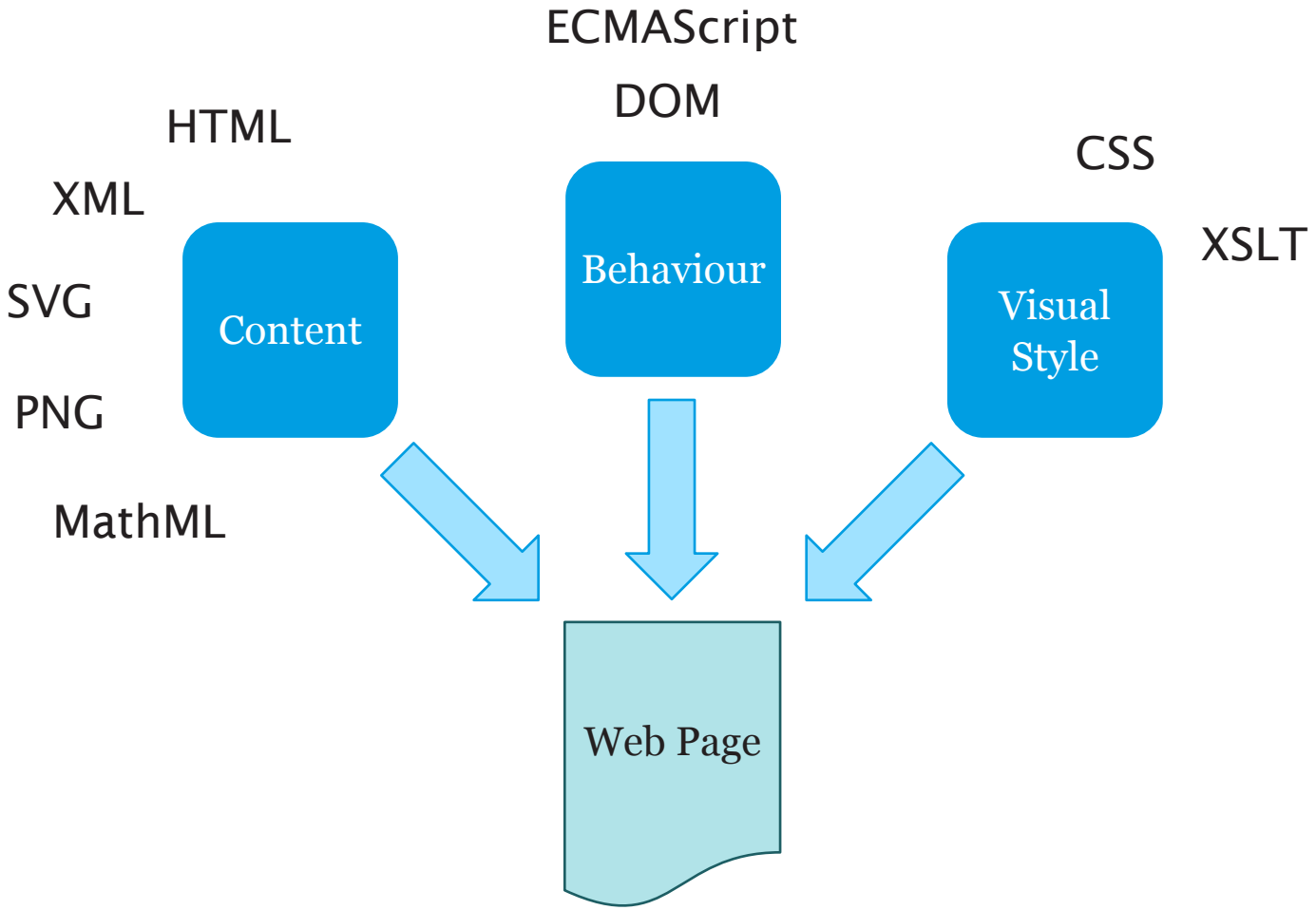
UNIVERSITY OF Southampton

Cascading Stylesheets

COMP3220 Web Infrastructure

Dr Nicholas Gibbins – nmg@ecs.soton.ac.uk

Content, Behaviour, Presentation



HTML style attribute

Presentation information specified in the style attribute of the relevant element

```
<!DOCTYPE html>
<html>
  <head>
    <title>Cinderella</title>
 </head>
  <body>
    <h1 style="color: red;">Cinderella</h1>
    <p>Once upon a time a wicked queen wanted to get
      rid of her pretty step-daughter.</p>
  </body>
</html>
```

Inline stylesheet

Presentation information provided inline in a style element

```
<!DOCTYPE html>
<html>
  <head>
    <style type="text/css">
      h1 { color: red; }
    </style>
  </head>
  <body>
    <h1>Cinderella</h1>
    <p>Once upon a time a wicked queen wanted to get
      rid of her pretty step-daughter.</p>
  </body>
</html>
```

External stylesheet

Presentation information stored in an external stylesheet, linked from the head of the document

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1>Cinderella</h1>
    <p>Once upon a time a wicked queen wanted to get
      rid of her pretty step-daughter.</p>
  </body>
</html>
```

CSS and XML

CSS stylesheets may also be applied to XML documents using a processing instruction

```
<?xml version="1.0"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.5//EN"
  "http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd">
<?xml-stylesheet type="text/css" href="book.css" ?>
<book>
  <chapter>
    <title>Cinderella</title>
    <para>Once upon a time a wicked queen wanted to get rid
      of her pretty step-daughter.</para>
  </chapter>
</book>
```

UNIVERSITY OF Southampton

Cascading Stylesheets

Why “Cascading” Stylesheets?

Stylesheets may be specified by different people:

- By the author of a web page
- By a user agent (a web browser)
- By a user

Cascading refers to the overriding of one stylesheet by another

The Cascade

In ascending order of precedence:

- User agent declarations (i.e. browser default)
- User normal declarations
- Author normal declarations
- Author important declarations
- User important declarations

Style rules have both importance and specificity

- Importance declared by the author of a stylesheet
- Specificity depends on the selectors in the style rules (see later)

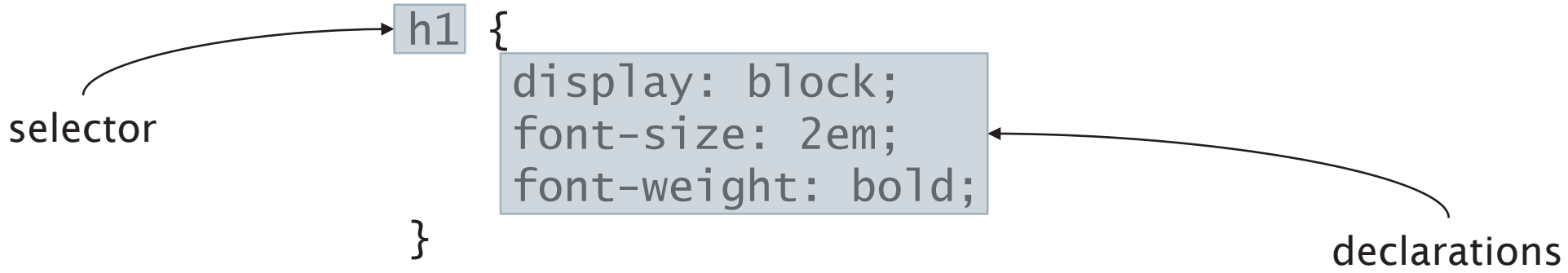
UNIVERSITY OF Southampton

Rule Sets

Anatomy of a CSS rule set

A rule set (also referred to as a rule) consists of:

- A selector
  (identifies the elements which are to be styled)
- A declaration block enclosed in curly braces, containing a series of declarations
  (specifies how the elements are to be styled)



Rule importance

A ruleset may be marked as important (see cascade precedence order)

- (avoid if possible)

```
h1 {
    font-size: 4em;
    text-decoration: blink
    ! important
}
```

@rules

`@import url("mystyle.css")`

- Includes the rule sets from another stylesheet

`@media` *`media-type`* `{ ... }`

- Specifies the target media type for the rulesets in the following block
- Key media types: `all`, `screen`, `print`, `speech`, `braille`

`@page { ... }`

- Used to define a page box for print media

`@font-face { font-family:` *`name`*`; src: url(...) ; }`

- Used to specify downloadable fonts – see later in this lecture

UNIVERSITY OF Southampton

Selectors

Basic selectors and combinators

`*`	Any element
`E`	An element of type E
`E#myid`	An E element with id="myid"
`E.warning`	An E element with class="warning"
`E F`	An F element that is a descendent of an E element
`E > F`	An F element that is a direct child of an E element
`E + F`	An F element that is immediately preceded by an E
`E ~ F`	An F element that is preceded by an E element

Attribute selectors

`E[foo]`	An E element with a foo attribute
`E[foo="bar"]`	An E element with a foo attribute whose value is bar
`E[foo^="bar"]`	An E with a foo attribute whose value starts with bar
`E[foo$="bar"]`	An E with a foo attribute whose value ends with bar
`E[foo*="bar"]`	An E with a foo attribute whose value contains bar

Pseudo-classes

`E:link`	An E element that is the anchor of an unvisited link
`E:visited`	An E element that is the anchor of a visited link
`E:active`	An E element that is being activated by the user
`E:hover`	An E element that is designated with a pointing device
`E:focus`	An E element with focus
`E:lang(fr)`	An E element in the language “fr”

Structural pseudo-classes

`E:root`	An E element that is the document root
`E:first-child`	An E element that is the first child of its parent
`E:last-child`	An E element that is the last child of its parent
`E:only-child`	An E element that is the only child of its parent
`E:first-of-type`	An E element that is the first E child of its parent
`E:last-of-type`	An E element that is the last E child of its parent
`E:empty`	An E element with no children

Pseudo-elements

`E::first-letter`	The first formatted letter of an E element
`E::first-line`	The first formatted line of an E element
`E::before`	Generated content before an E element
`E::after`	Generated content after an E element

Selector specificity

Specificity is based on (in decreasing order of importance):

- Whether the declaration appears in a style attribute
- The number of ID attributes in the selector
- The number of other attributes and pseudo-classes in the selector
- The number of elements and pseudo-elements in the selector

Specificity example

In increasing order of specificity:

```
*                   { ... }
li                  { ... }
li:first-line       { ... }
ul li               { ... }
ul ol + li          { ... }
h1 + [*rel=up]      { ... }
ul ol li.red        { ... }
li.red.level        { ... }
#x34y               { ... }
style="..."
```

Example

What colour is the p element?

```
<!DOCTYPE html>
<html>
  <head>
    <style type="text/css">
      p { color: blue }
      #x97z { color: red }
    </style>
  </head>
  <body>
    <p id="x97z"
       style="color: green">SOME TEXT</p>
  </body>
</html>
```

UNIVERSITY OF Southampton

Declarations

Declarations

Declarations are a series of property-value pairs

- Separated by ";"

Many properties!

- Too many to list exhaustively in a lecture (but I'll show the most common ones)

Common properties

```
font-family: Arial;
font-size: 32pt;
font-style: italic;
font-weight: bold;
line-height: 14pt;
color: green;
color: rgb(0,128,0);
background-color: #ffff00;
opacity: 0.8;
text-align: center;
text-transform: uppercase;
text-indent: 3em;
text-decoration: blink;
```

```
display: block;
margin: 12pt 12pt 12pt 12pt;
border: 0.1in solid red;
border-right: 1cm;
border-top-color: green;
padding: 6pt 6pt 6pt 6pt;
width: 80%;
float: left;
clear: left;
direction: rtl;
```

Fonts

`font-family:`

- Specifies a typeface by name
  e.g. Eurostile, Arial, Alberto
- Some generic typefaces:
  `serif, sans-serif, serif, monospace, cursive`
- Specifies a list of typefaces to try in order
  e.g. `Freight, Georgia, serif`

`font-style:`

- Specifies a font style within a typeface
  e.g. `normal, italic, oblique`

`font-weight:`

- Specifies how heavy a font is
  e.g. `normal, bold, bolder`

`font-size:`

- Specifies how big a font is
- Absolute size:
  - `px` – pixels
  - `pt` – points (1/72 of an inch)
- Relative size (to parent element):
  - `%` - percentage
  - `em` - width of lower-case “m”

Web Fonts

@font-face used to specify a downloadable font (usually in .woff format)

```
@font-face {
  font-family: Gentium;
  src: url(http://example.com/fonts/Gentium.woff);
}

p {
  font-family: Gentium, serif;
}
```

Colours

`color:`

- Text colour

`background-color:`

- Colour of element's background

`opacity:`

- How opaque is the element? (number between 0 and 1 or percentage)

Colour specifications

- by name: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, and yellow
- by hex value
  `#rgb`
  `#rrggbb`
  e.g. `#ff0000` is red
- by rgb value
  `rgb(255, 0, 0)`
  `rgb(100%, 0%, 0%)`

Backgrounds

`background-image: url(...)`

- Specifies an image to use as the background for an element

Text rendering

`text-indent:`

- Specifies indentation of first line of an element
  `text-indent: 3em;`
  `text-indent: 10%;`

`text-align:`

- `left, right, center, justify`

`text-decoration:`

- `none, underline, overline, line-through, blink`

`text-transform:`

- `none, capitalize, uppercase, lowercase`

`letter-spacing:`

- Adjusts spacing between characters
  e.g. `letter-spacing: 0.5em;`

`word-spacing:`

- Adjusts spacing between words
  e.g. `word-spacing: 1em;`

`line-height:`

- `Adjusts spacing between consecutive lines (baseline to baseline)`
  `e.g. line-height 14pt`

Generated content

CSS can be used to automatically add content to elements

- Used in conjunction with the `:before` and `:after` pseudo-elements
- Commonly used for adding quotation marks and numbering

Special values:

- `open-quote, close-quote`
- `attr(X)`
  Returns the value of attribute X on the element

```
p.note::before {
  content: “Note: ”;
}

q::before {
  content: open-quote;
}

q::after {
  content: close-quote;
}
```

Generated content: numbering

`counter(...)`

- Returns the value of a counter variable

`counter-increment:`

- Adds one to a counter variable

`counter-reset:`

- Resets the value of a counter variable to 0

```
body {
  counter-reset: chap;
}

h1:before {
  content: “Chapter ” counter(chap) “: ”;
  counter-increment: chap;
}

h1 {
  counter-reset: sect;
}

h2:before {
  content: counter(chap) “.”
           counter(sect) “ ”;
  counter-increment: sect;
}
```

Inheritance

A property may have a special value of `inherit`

- For a given element, the property takes the same value as the same property on the element's parent

```
body {
  color: black;
  background-color: white;
}

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

UNIVERSITY OF Southampton

CSS Visual Formatting Model

Visual Formatting Model

Central underlying model for CSS is of a hierarchy of nested boxes

- Normal flow (using the box model)
- Also: floats and absolute positioning

Some variant models under development:

- Multi-Column (WD, 15 Oct 2019 – has already been to CR and then dropped back to WD)
- Flexible Box (CR, 19 Nov 2018 – in CR since 2016)
- Box Alignment (WD, 21 Apr 2020 – in WD since 2012)
- Grid (CR, 18 Aug 2020 – in development since 2012, and has been to CR once before)

Visual Formatting Model

`display:`

- Indicates how an element is to be nested within its parent (alternatively, what type of box it is to be generated for it)

`display: block`

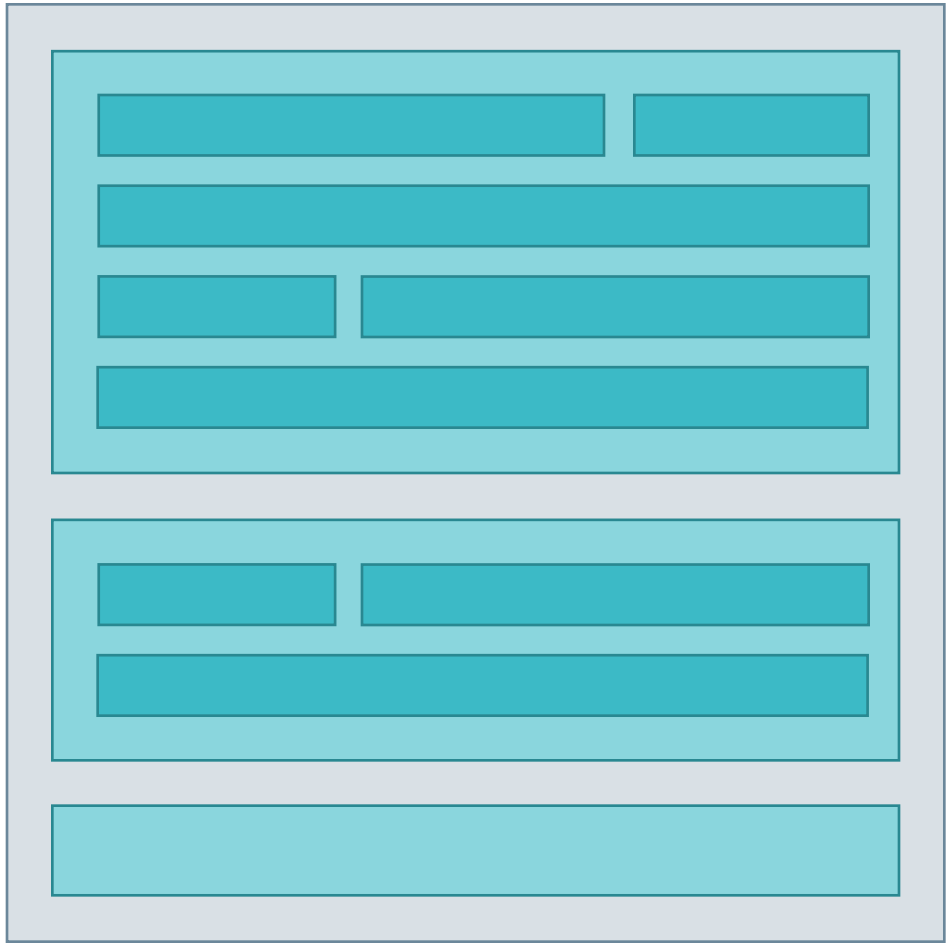
- Generate a block box
- Used for div-like elements

`display: inline`

- Generate an inline box
- Used for span-like elements

`display: none`

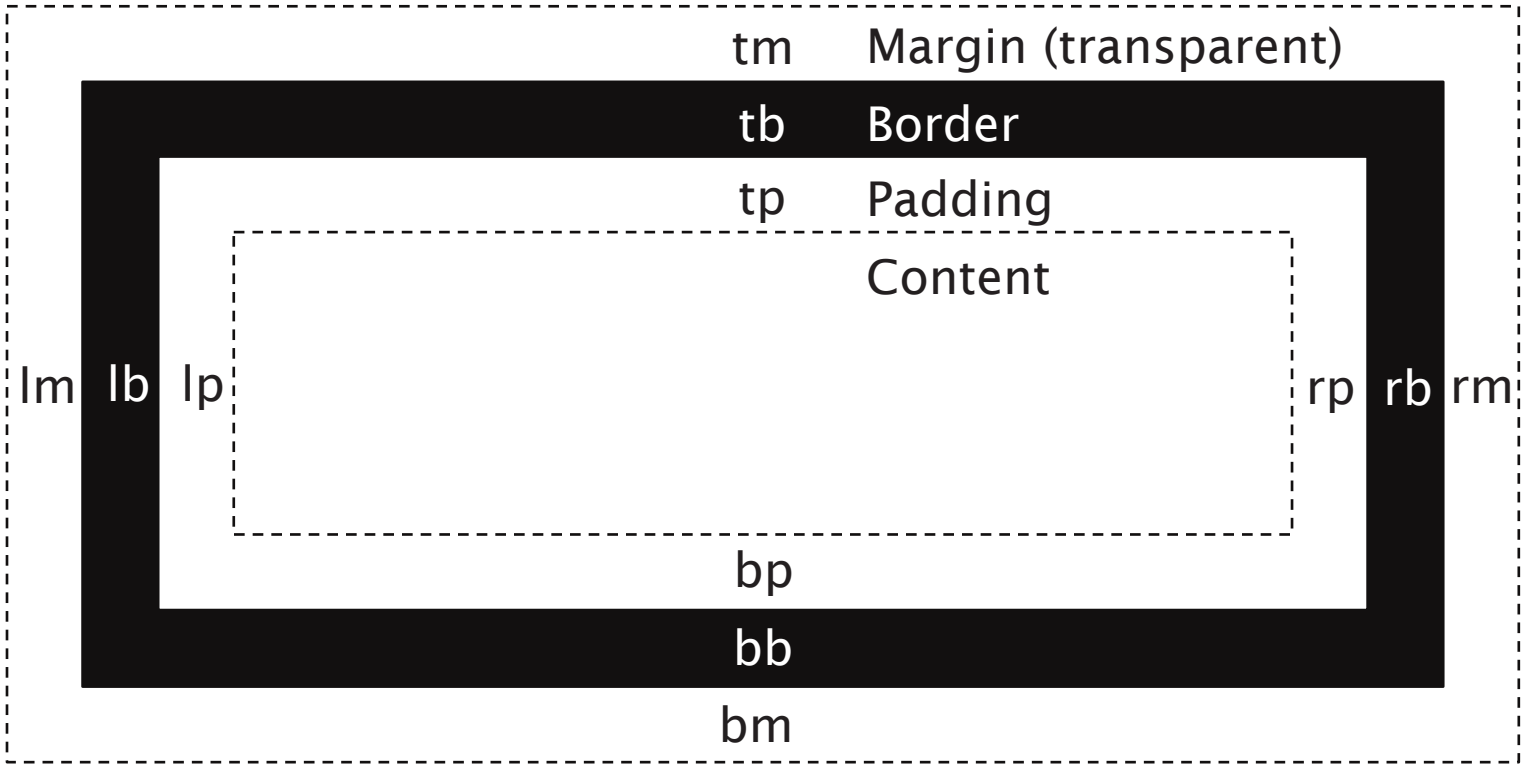
- No box is generated (element is not displayed)



The Box Model

Defines box in terms of margin, border and padding

- Separate values for top, right, bottom and left



Box Model properties

`margin-top: margin-right: margin-bottom: margin-left:`

`padding-top, padding-right, padding-bottom, padding-left`

- Specify width of padding/margin (`px, pt, em`)

`border-top-width: border-right-width:`
`border-bottom-width: border-left-width:`

- `thin, medium, thick`
- `px, pt, em`

`border-top-color: border-right-color:`
`border-bottom-color: border-left-color:`

- Specified as for `color:` and `background-color:`

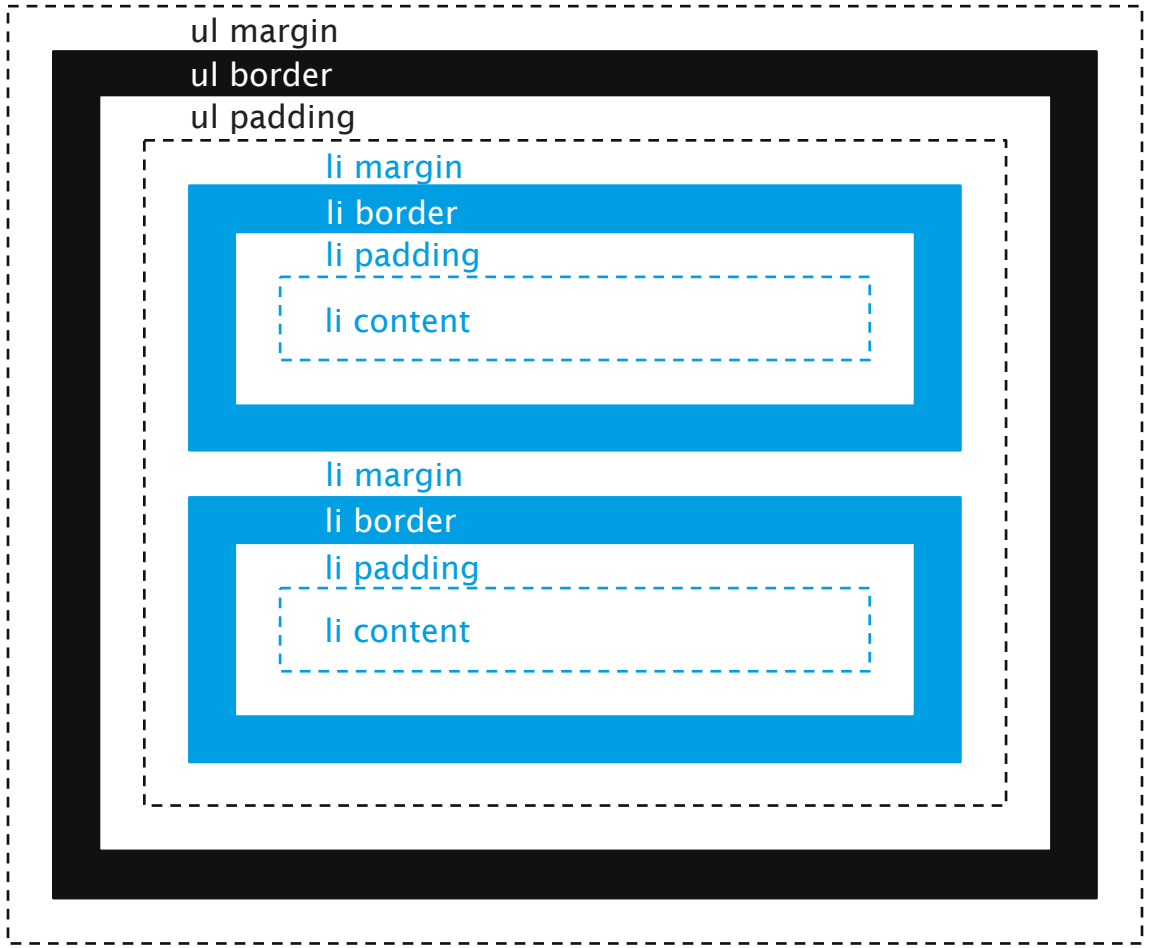
`border-top-style: border-right-style:`
`border-bottom-style: border-left-style:`

- `none, dotted, dashed, solid, double, groove, ridge, inset, outset`

Collapsing Margins

Adjoining vertical margins combine to form a single margin

```
<ul>
  <li>...</li>
  <li>...</li>
</ul>
```



Floating elements

A floated element can move left/right (within its containing block), allowing content to flow around it

`float:`

- `left, right, none`

`clear:`

- Indicates that an element's content may not flow around a floated element
- `left, right, both`

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras et feugiat lorem. Fusce congue lacinia ante ac tincidunt. Donec et metus nec orci dictum ultricies. Nam malesuada justo justo, ac volutpat lectus commodo lacinia. Mauris vitae nulla nisi. Ut dignissim pellentesque est vel maximus. Phasellus vestibulum quam sollicitudin mauris egestas,

Floating elements

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <img style="float: left"
         src="kitten.jpg"/>
    <p>Lorem ipsum dolor sit amet,
       consectetur adipiscing
       elit. Cras et feugiat
       lorem [...]</p>
  </body>
</html>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras et feugiat lorem. Fusce congue lacinia ante ac tincidunt. Donec et metus nec orci dictum ultricies. Nam malesuada justo justo, ac volutpat lectus commodo lacinia. Mauris vitae nulla nisi. Ut dignissim pellentesque est vel maximus. Phasellus vestibulum quam sollicitudin mauris egestas,

Positioning

Elements can be positioned outside the normal flow

`position: static`

- Normal flow applies to element box

`position: relative`

- Box is displaced relative to its normal (static) position

`position: absolute`

- Box is positioned relative to its containing box

`position: fixed`

- Box is positioned relative to the viewport (screen) or page

`top: right: bottom: left:`

- Used to specify offsets for use with the relative, absolute and fixed positioning schemes

A note on standardisation

Very many W3C CSS standards documents:

- Recommendations: 15
- Notes: 7
- Candidate Recommendations: 28
- Drafts: 68

Very large number of standards documents

- Varying degrees of maturity
- Varying degrees of adoption

Future* CSS

- Other layouts: Grid, Flexible Box, Multi-column, ...
- Animation
- Drop shadows, rounded borders
- Custom counters (for headings and lists)
- Text wrapping around shapes
- Web fonts
- Many others...

* Possibly usable now, but not yet standardised

Further reading

List of W3C CSS standards and standards-in-progress

- https://www.w3.org/standards/techs/css

CSS Zen Garden

- A demonstration of CSS capabilities – many stylings of the same HTML page
- http://www.csszengarden.com/

UNIVERSITY OF Southampton

Next Lecture: XSLT