



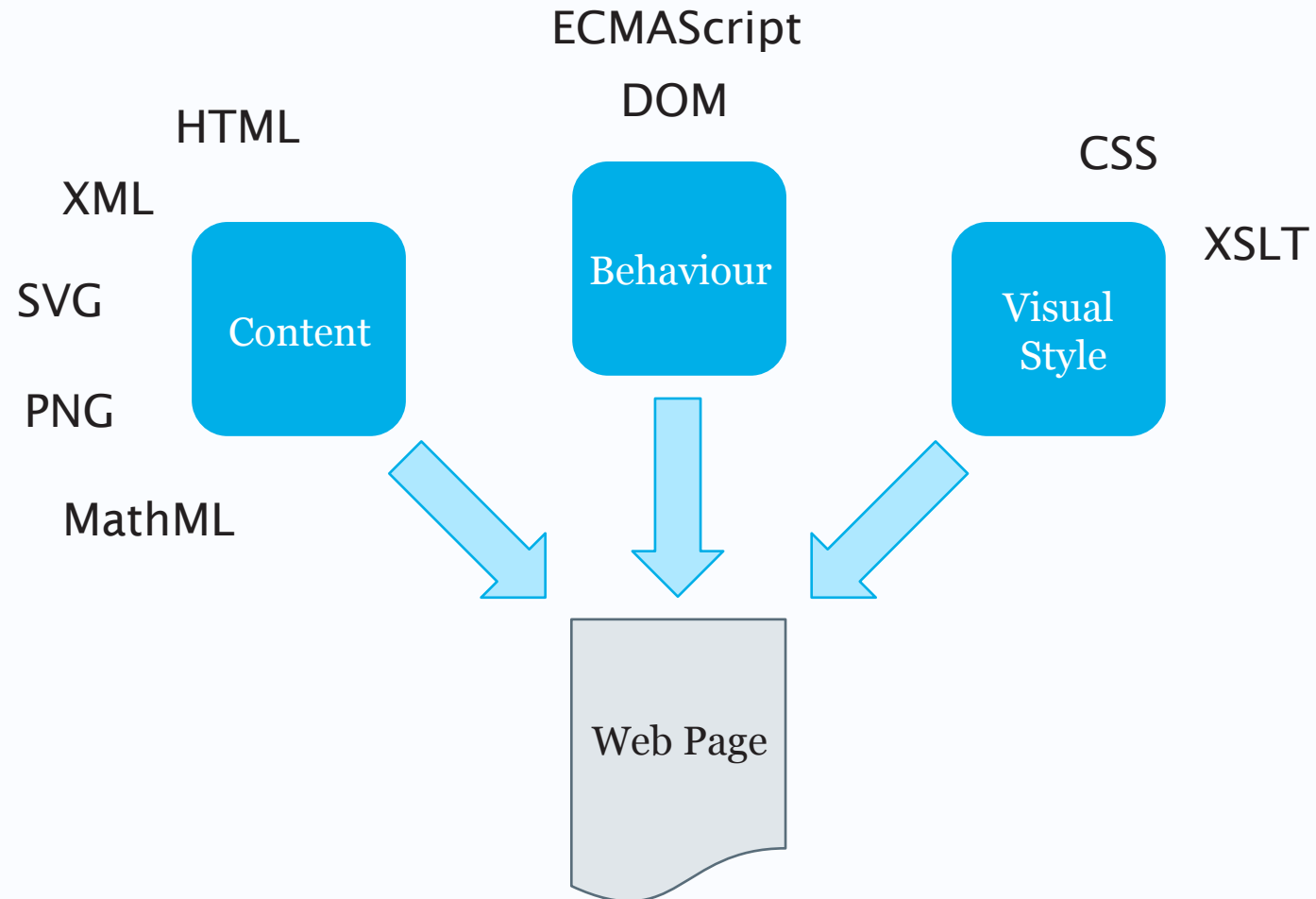
University of
Southampton

Styling the Web

COMP3227 Web Architecture & Hypertext Technologies

Dr Heather Packer – hp3@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>
```

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**

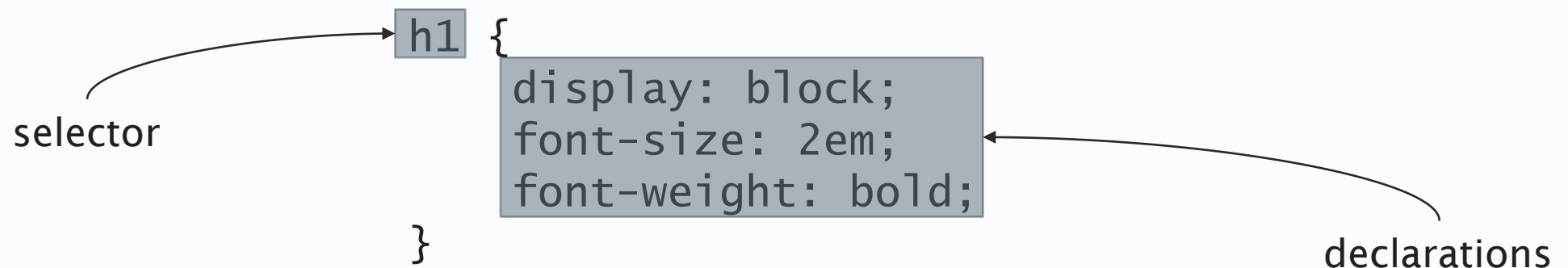
- Importance declared by the author of a stylesheet
- Specificity depends on the selectors in the style rules (see later)

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

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

<code>E[foo]</code>	An E element with a foo attribute
<code>E[foo="bar"]</code>	An E element with a foo attribute whose value is bar
<code>E[foo^="bar"]</code>	An E with a foo attribute whose value starts with bar
<code>E[foo\$="bar"]</code>	An E with a foo attribute whose value ends with bar
<code>E[foo*="bar"]</code>	An E with a foo attribute whose value contains bar

Pseudo-classes

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

Structural pseudo-classes

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

Pseudo-elements

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

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;
}
```

# 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`

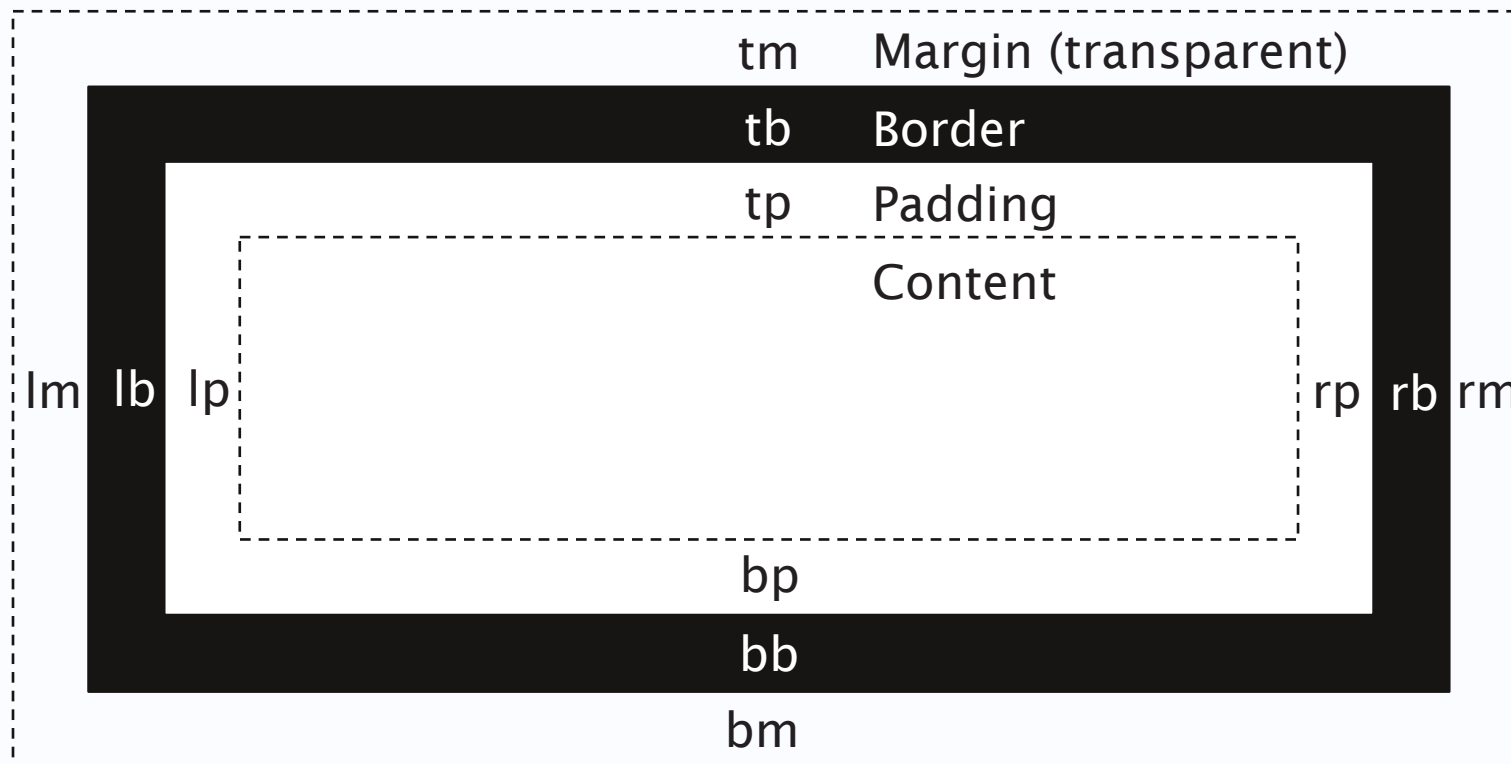
- No box is generated (element is not displayed)



# 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

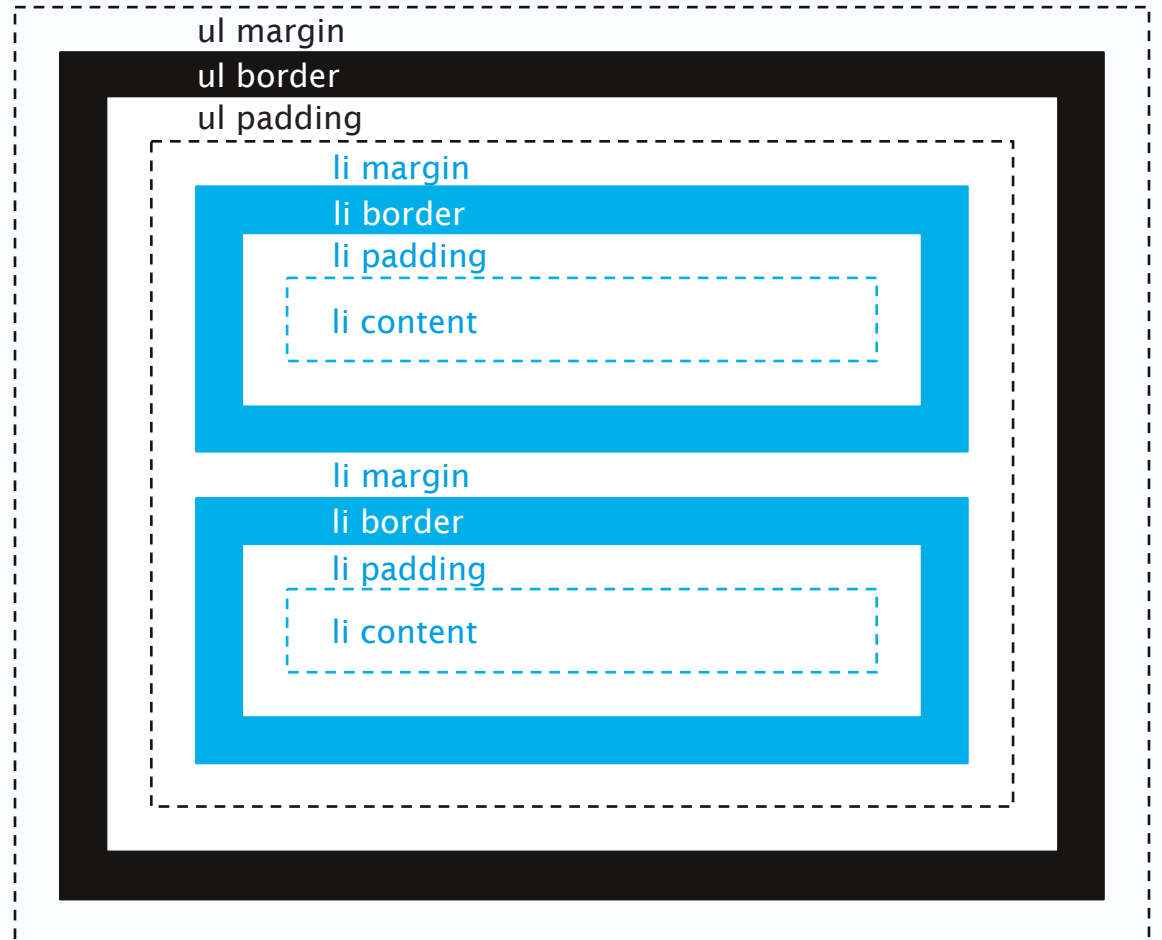
# Collapsing Margins

Adjoining vertical margins combine to form a single margin

```

 ...
 ...

```





# 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>

 <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/>

# eXtensible Stylesheet Language

# The Problem

How do you maintain different versions of a document for presentation on different systems that:

- Have different presentation characteristics (screen size, etc)?
- Require different document formats?

XSL is a family of XML-based technologies designed to address this problem



# eXtensible Stylesheet Language

## XSL Transformations (XSLT)

- XML-based language for describing transformations from one XML-based language to another

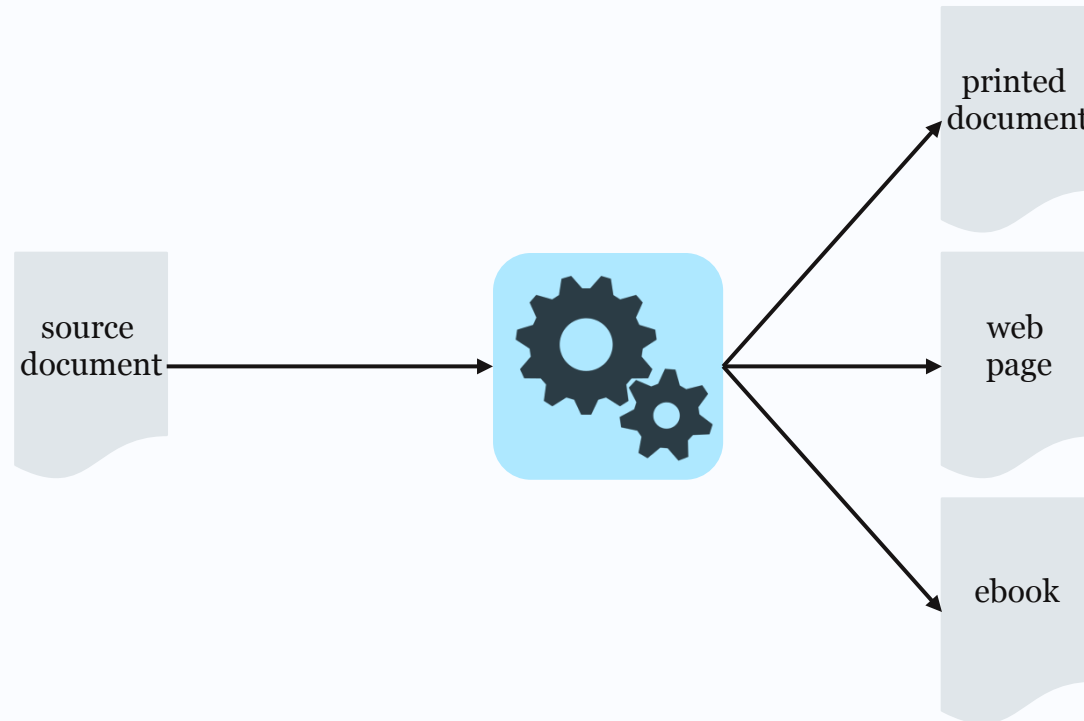
## XML Path Language (XPath)

- Language for referring to parts of an XML document

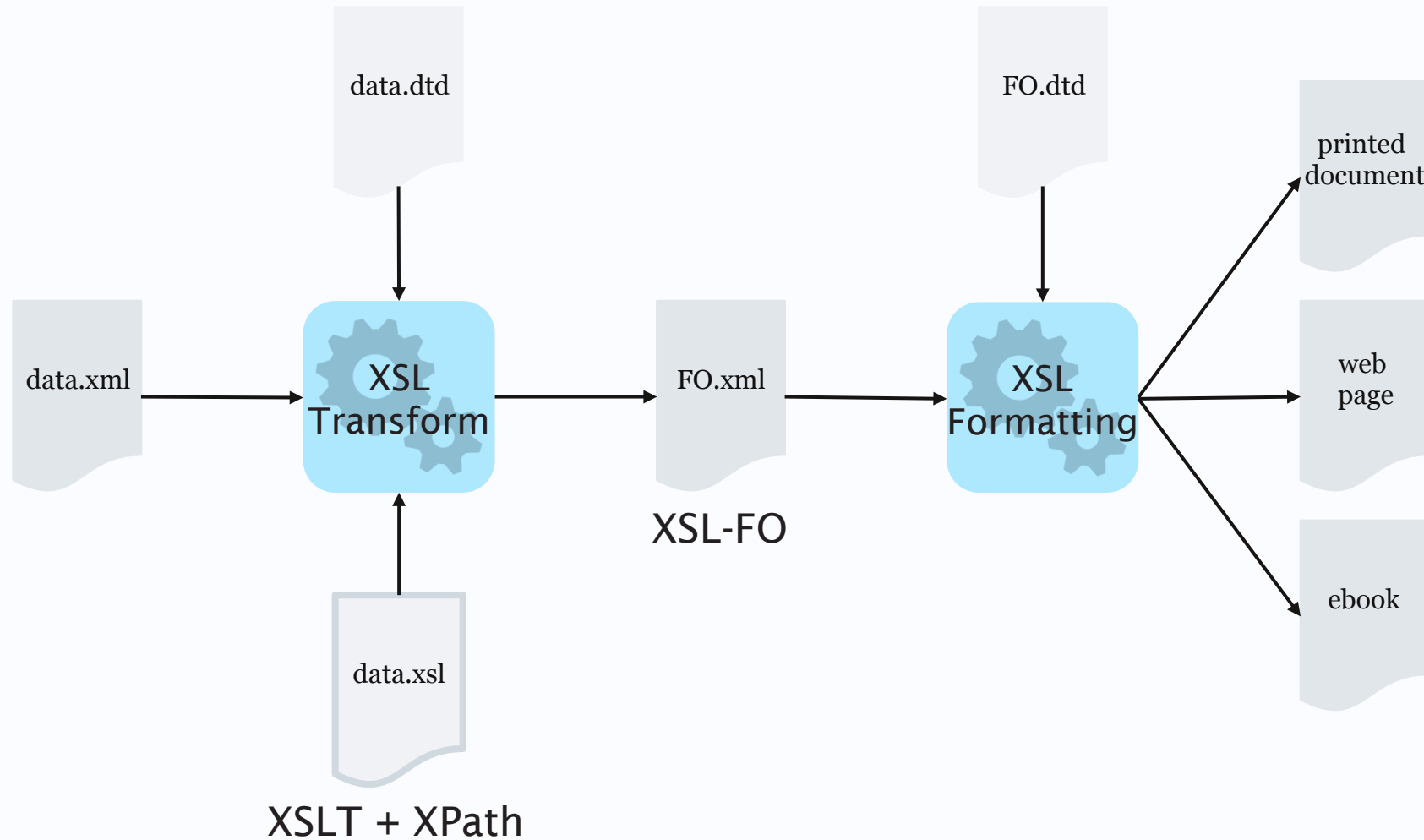
## XSL Formatting Objects (XSL-FO)

- XML vocabulary for specifying formatting semantics

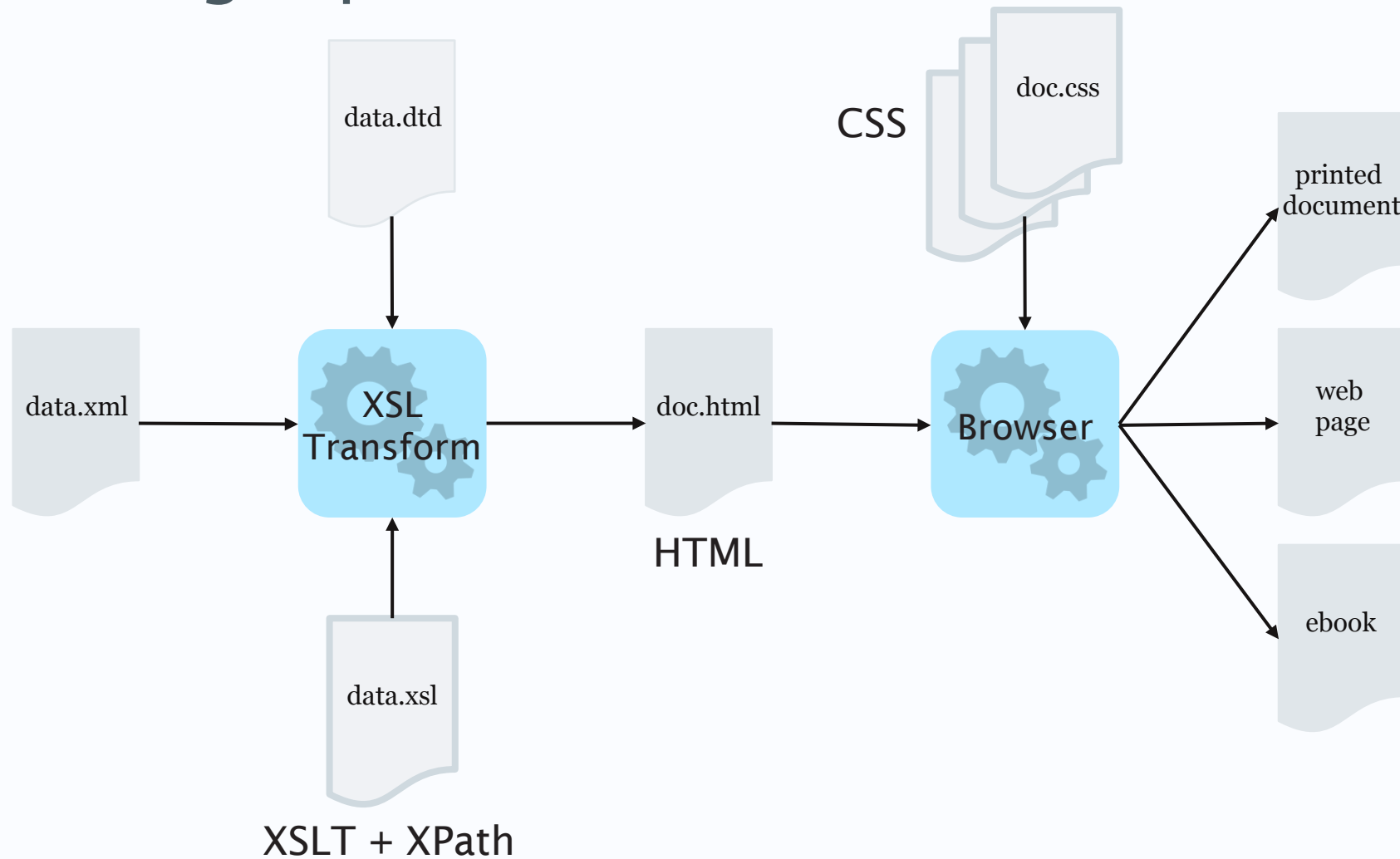
# XML processing



# XSL processing in theory



# XSL processing in practice



# Example input

```
<?xml version="1.0" encoding="UTF-8"?>
<contacts>
 <name>My address list</name>
 <card>
 <name>Fred Foo</name>
 <email>fred@example.org</email>
 <tel type="work">01234567890</tel>
 </card>
 <card>
 <name>Jill Bar</name>
 <email>jill@example.org</email>
 <tel type="home">02345678901</tel>
 </card>
</contacts>
```

# Example output

```
<html xmlns="http://www.w3.org/TR/xhtml1/strict">
 <head>
 <title>My address list</title>
 </head>
 <body>
 <h1>My address list</h1>
 <div>
 <h2>Fred Foo</h2>
 <p>Email:
 fred@example.org</p>
 <p>Telephone: 01234567890</p>
 </div>
 <div>
 <h2>Jill Bar</h2>
 <p>Email:
 jill@example.org</p>
 <p>Telephone: 02345678901</p></div>
 </body>
</html>
```

# XSL stylesheets

An XSL stylesheet consists of a number of templates

Each template:

- Matches an element in the original document using XPath
- Specifies the new content with which the element is to be replaced

# XPath

Expression language for specifying nodes (elements, attributes) within an XML document

Specifies nodes using a path through the hierarchy of the document

- Can be absolute (from the root) or relative (from current position)



# XPath expressions

E	Selects all nodes with the name E
/	Selects from the root node
//	Selects nodes anywhere under the current node
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

# XPath path expression examples

<code>contacts</code>	Selects all contacts elements
<code>/contacts</code>	Selects the root contacts element
<code>contacts/card</code>	Selects all card elements that are children of contacts
<code>//card</code>	Selects all card elements
<code>contacts//name</code>	Selects all name elements that are descendants of contacts
<code>//tel/@lang</code>	Selects type attribute on all tel elements

## xsl:template

match attribute contains XPath expression that identifies an element or elements

- Content of element is either output markup, or other XSL directives

```
<xsl:template match="/">
```

```
...
```

```
</xsl:template>
```

## xsl:apply-templates

Used within body of a template

- Recursively applies templates to children of the element
- `select` attribute identifies target child node using XPath

```
<xsl:apply-templates select="..." />
```

## xsl:for-each

Loops over every matching element

- select attribute identifies target node set

```
<xsl:for-each select="...">
```

```
...
```

```
</xsl:for-each>
```

## xsl:value-of

Extracts the value of an XML element and uses it in the output document

- select attribute used to identify element

```
<xsl:value-of select="..." />
```

# XSLT example – input

```
<?xml version="1.0" encoding="UTF-8"?>
<contacts>
 <name>My address list</name>
 <card>
 <name>Fred Foo</name>
 <email>fred@example.org</email>
 <tel type="work">01234567890</tel>
 </card>
 <card>
 <name>Jill Bar</name>
 <email>jill@example.org</email>
 <tel type="home">02345678901</tel>
 </card>
</contacts>
```

# XSLT example – output

```
<html xmlns="http://www.w3.org/TR/xhtml1/strict">
 <head>
 <title>My address list</title>
 </head>
 <body>
 <h1>My address list</h1>
 <div>
 <h2>Fred Foo</h2>
 <p>Email:
 fred@example.org</p>
 <p>Telephone: 01234567890</p>
 </div>
 <div>
 <h2>Jill Bar</h2>
 <p>Email:
 jill@example.org</p>
 <p>Telephone: 02345678901</p></div>
 </body>
</html>
```



# XSLT example – using stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="contacts.xsl" version="1.0"?>
<contacts>
 <name>My address list</name>
 <card>
 <name>Fred Foo</name>
 <email>fred@example.org</email>
 <tel>01234567890</tel>
 </card>
 <card>
 <name>Jill Bar</name>
 <email>jill@example.org</email>
 <tel>02345678901</tel>
 </card>
</contacts>
```

# XSLT example – stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns="http://www.w3.org/TR/xhtml1/strict">
```

...

```
</xsl:stylesheet>
```

# XSLT example – templates

```
<xsl:template match="tel">
 <p>Telephone: <xsl:value-of select="."/></p>
</xsl:template>
```

# XSLT example – templates

```
<xsl:template match="email">
 <p>Email: <xsl:value-of select="."/></p>
</xsl:template>
```

# XSLT example – templates

```
<xsl:template match="name">
 <h2><xsl:value-of select="."/></h2>
</xsl:template>
```

# XSLT example – templates

```
<xsl:template match="/">
 <html>
 <head>
 <title><xsl:value-of select="/contacts/name"/></title>
 </head>
 <body>
 <h1><xsl:value-of select="/contacts/name"/></h1>
 <xsl:for-each select="/contacts/card">
 <div>
 <xsl:apply-templates select="name"/>
 <xsl:apply-templates select="email"/>
 <xsl:apply-templates select="tel"/>
 </div>
 </xsl:for-each>
 </body>
 </html>
</xsl:template>
```

# XSL versus CSS

Considering CSS...

## Pros

- Simple (relatively)
- Cascading recognises different needs of users and authors

## Cons

- Unable to modify document structure

# XSL versus CSS

Considering XSL:

## Pros

- Able to modify and transform document structure
- Better for data

## Cons

- Complex (relatively)
- Cumbersome for ordinary documents
- No consideration of differing needs of users versus authors



## Further reading

- Kay, M. (2017) *XSL Transformations Version 3.0*. W3C Recommendation. Cambridge, MA: World Wide Web Consortium.
  - <https://www.w3.org/TR/2017/REC-xslt-30-20170608/>

**Next:**  
**Optimising the Web**