An Introduction to Cascading Style Sheets

Laboratory of Computer Technologies L-A

Laboratorio di Tecnologie Informatiche L-A

Andrea Omicini & Giulio Piancastelli
{andrea.omicini, giulio.piancastelli}@unibo.it

Ingegneria Due
Alma Mater Studiorum—Università di Bologna a Cesena

Academic Year 2007/2008

Web Style Sheets

- Style sheets for the Web aims at describing how elements in a document must be presented on a set of heterogeneous media, such as paper, video, audio, medium for disabilities
- Style sheets allow separating style description from content and its structure

Style sheets are a World Wide Web Consortium (W3C) Standard described at `http://w3.org/Style/`. The standard is comprised by two languages:

Cascading Style Sheet (CSS), composed by two specifications called CSS Level 1 (CSS1) and CSS Level 2 Revision 1 (CSS2.1)

Extensible Stylesheet Language (XSL) composed by three specifications called XSL Transformations (XSLT), XML Path Language (XPath), XSL Formatting Objects (XSL-FO)

Why Two Languages?

The CSS language

- can be used with HTML and XML
- but has its own syntax, and is not general enough to be a transformation language

The XSL language

- is a transformation language, e.g. can be used to transform an XML page into a HTML/CSS document
- features an XML syntax
- can be used with XML only, not with HTML

CSS and XSL share the same underlying formatting model and can also be used together

CSS Specifications

CSS1 is a W3C Standard describing a simple style sheet mechanism that allows authors and readers to attach style (e.g. fonts, colors and spacing) to HTML documents

CSS2.1 is a W3C Standard adding support for media-specific style sheets, content positioning, table layout, internationalization features, and properties related to the user interface

CSS3 is still a W3C Working Draft

CSS Level 1 Specification

CSS1 specification can be found at `http://www.w3.org/TR/CSS1`

Other specifications can be found on the W3C website

CSS Rules

A style sheet consists in one or more rules that control how selected elements should be displayed

Definition

A CSS rule is composed of two parts: a *selector* and one or more *declarations* enclosed in curly braces; each declaration is composed by a *property* and a *value*. The general form is:

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

Example

The CSS rule

```
h1 { color: blue; }
```

selects `h1` elements in a document and applies the style declaration by which their text is displayed in blue.

Selectors

A simple selector is a selector that matches an element based on its type and/or attributes. Simple selectors are of 3 types:

- HTML elements
- `class` attribute values
- `id` attribute values

Classes as selectors

Classes are user defined names to group elements by means of the `class` attribute. Classes are used as selectors as follows:

```
.footnotes { font-size: x-small; }
```

ID as selectors

ID are user defined names to uniquely identify specific elements by means of the `id` attribute. ID are used as selectors as follows:

```
#photo { margin: 0; }
```

Combining Selectors (I)

Simple selectors can be combined to define more precise rules:

```
/* Examples of simple selector combinations */
div.warning { color: red; }
img.screenshot { border: 1px solid black; }
p#copyright { font-size: 80%; }
table#results { background-color: #666; }
```

Only one class or one id can be specified per simple selector

Pseudo-elements and pseudo-classes

Selectors can be further detailed by using *pseudo-elements* and *pseudo-classes*

Pseudo-elements

A pseudo-element is a part of an element that does not correspond to a real element in the source document, but corresponds to a separate display object that can be styled by itself.

First letter and first line

CSS1 defines 2 pseudo-elements: the `first-letter` of an element and the `first-line` as it appears on the display

```
div:first-letter {
    font-size: 200%;
    text-transform: uppercase;
}

p:first-line { font-weight: bold; }
```

Only a restricted set of properties can be applied to each pseudo-element

Pseudo-classes

Pseudo-classes reflect the fact that the same element must sometimes be given different styling, depending on external information not found in the document.

Hyperlink

A hyperlink is usually displayed in a different style after the user has visited the target, even though nothing in the source document has changed.

Example

CSS1 defines 3 pseudo-classes for anchors with a `href` attribute:

```
a:link { color: #ff0000; }
a:visited { color: #00f; }
a:active { color: rgb(0, 255, 0); }
```

The `link` pseudo-class is for unvisited links and `visited` is for visited links. The `active` pseudo-class applies while a link is being selected by the user, e.g. through a mouse button press.

Combining Selectors (II)

Pseudo-elements and pseudo-classes can be combined with other selectors:

```
/* More examples of selector combinations */
p#initial:first-letter { color: red; }
a.external:visited { text-decoration: none; }
```

Pseudo-elements combinations

There are some restrictions using pseudo-elements:

- when combined with classes or pseudo-classes, pseudo-elements must be specified at the end of the selector
- only one pseudo-element can be specified per selector

Pseudo-classes combinations

Normal class names precede pseudo-classes in the selector

Contextual Selectors

Alongside simple selectors, CSS1 defines a first elementary form of contextual selectors.

Definition

A contextual selector is a selector that matches an element based on its position in the document structure. A contextual selector consists of several simple selectors separated by whitespace.

Example

The following CSS rule

```
li em { text-decoration: underline; }
```

matches any `em` element that is a descendant of a `li` element, displaying its text as underlined.

Restrictions

In a contextual selector, pseudo-elements are only allowed at the end of the selector

Grouping Selectors

Selectors can be grouped in comma-separated lists:

```
h1, h2, .definition, div.ingress p:first-line, #name {
    font-variant: small-caps;
}
```

Grouping is typically allowed to reduce the size of the style sheet

Grouping Properties

Shorthand properties allow several values to be set in one declaration. The following rule, using the `font` property

```
p { font: bold 10px/14px "Gill Sans", Arial, sans-serif }
```

is equivalent to

```
p {
    font-style: normal;
    font-variant: normal;
    font-weight: bold;
    font-size: 10px;
    line-height: 14px;
    font-family: "Gill Sans", Arial, sans-serif;
}
```

Shorthand properties are defined for specific set of properties, including `margin`, `border`, and `padding`

Property Inheritance

Most style properties of any element are inherited from its parent element. Given the style:

```
p { font-size: 10px; }
```

in the following example:

```
<p>Hello <em>CSS</em> world!</p>
```

if the `em` elements are not given a font size, the `em` element child of the `p` element inherit the `10px` font size value

Inheritance does not always hold

Some style properties are not inherited from the parent element to the child element. For example, the `background` property does not inherit.

Default Property Values

Some properties have a default value defined by the W3C Standard

Setting default styles for a document

To set a "default" style property for a document, that property can be set on an element from which all visible elements descend. In HTML documents, the body element can serve this function:

```
body {
    color: black;
    background: url(texture.gif) white;
}
```

The example above sets the text color to be black and the background to be an image. The background will be white if the image is not available.

Percentage Properties

Often, the value of a property is a percentage that refers to another property. For each property that allows percentage values, it is defined what property it refers to.

Example

In the following percentage example, `line-height` refers to the `font-size` property value:

```
p {
    font-size: 10pt;
    /* relative to font-size, i.e. 12pt */
    line-height: 120%;
}
```

and children elements of `p` will inherit the computed value of `line-height` (12pt), not the percentage

Applying Styles to Web Pages

Style sheets can be applied to a web page in any of three ways: *external*, *embedded*, or *inline*

```
<html>
<head>
  <link rel="stylesheet" href="/styles/style.css"
        type="text/css" media="screen">
  <title>Applying Styles to Web Pages</title>
  <style type="text/css" media="all">
  @import url("/styles/additional.css");
  h1 { color: blue; }
  </style>
</head>
<body>
  <h1>This is a h1 element, displayed in blue</h1>
  <p style="color: green;">A paragraph, displayed in green.</p>
</body>
</html>
```

Formatting Model

There are two kinds of elements in HTML documents:

- **inline** do not have a "newline" after and before, as `span`, `strong`, `em`, ...
- **block** are displayed on a line on their own, as headers, paragraphs, list elements, ...

The `display` property

The CSS property defining the inline or block behavior is called `display`, and assumes values such as `inline`, `block`, `none`. The following example

```
li { display: inline; }
```

shows list elements on a single line instead of a line each

Restrictions

Pseudo-elements can only be applied to block-level elements

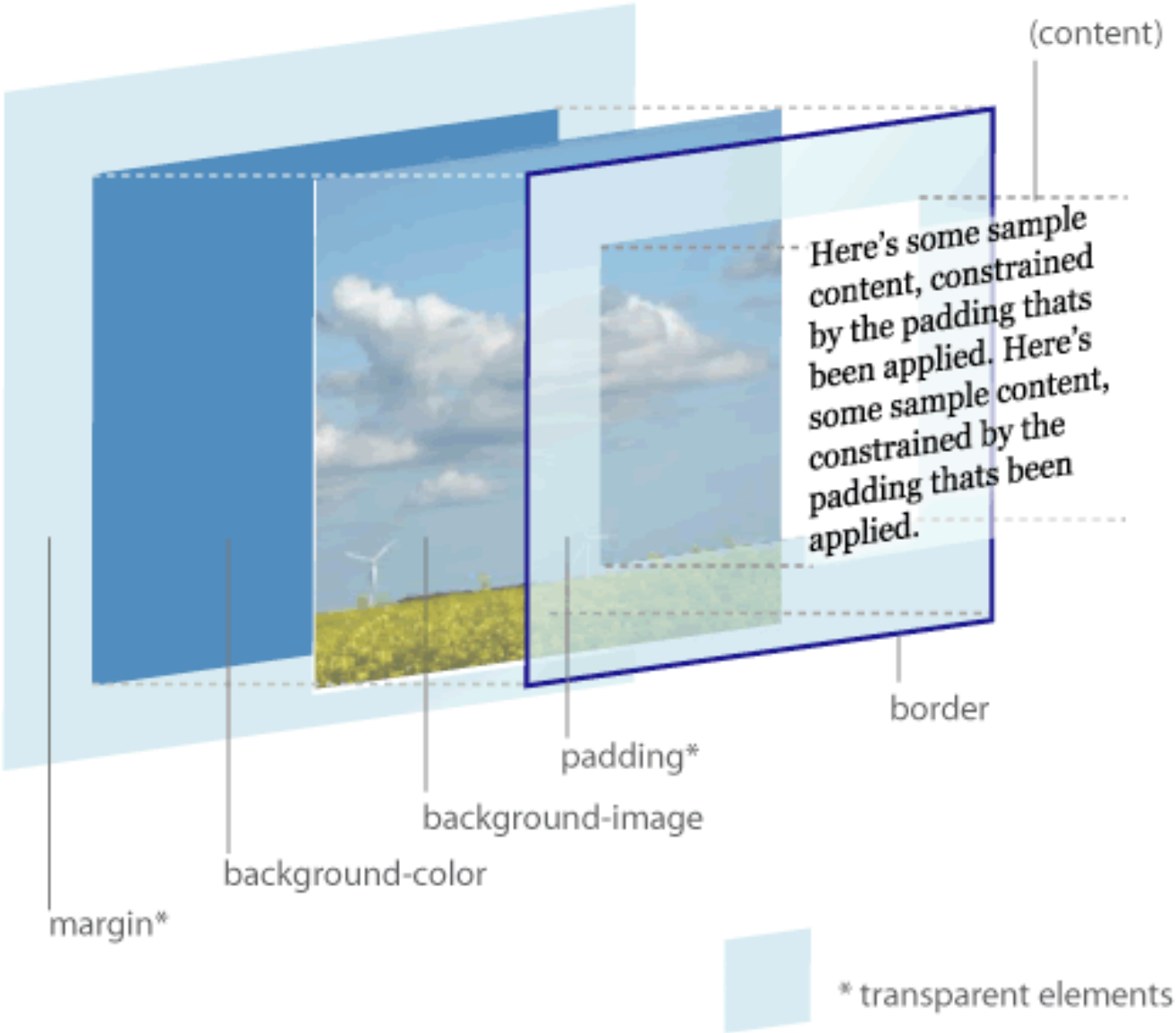
The Box Model (1/2)

The box model describes the rectangular boxes that are generated for each element in the document and laid out according to the formatting model.

- Each element is composed by a box enclosing its **content**
- The content is constrained by a **padding** box
- The padding separates content from its **border**
- The border enclose the element **background**
- The background is constrained by a **margin** box

Shorthand properties for tuning the parts of each box are provided: `padding`, `border`, `background`, `margin`

The Box Model (2/2)



Floating Elements

Using the `float` property, an element can be declared to be outside the normal flow of elements and is then formatted as a block-level element.

Example

For example, by setting the `float` property of an image to `left`:

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

the image is moved to the left until the margin, padding or border of another block-level element is reached. The normal flow will wrap around on the right side. The margins, borders and padding of the element itself will be honored, and the margins never collapse with the margins of adjacent elements.

Cascading Style Sheets

CSS are said to be *cascading* because more than one style sheet can influence the presentation at the same time. There are two main reasons for the cascading feature:

modularity a style sheet designer can combine several partial style sheets to reduce redundancy

```
@import url(http://example.com/pastoral);
@import url(http://example.com/marine);
h1 { color: red }
```

author/reader balance both readers and authors can influence the presentation through style sheets using the same language, thus reflecting a fundamental feature of the web: everyone can become a publisher.

Sometimes conflicts will arise between the style sheets that influence the presentation. Conflict resolution is based on each style rule having a weight.

The important Overruling

Style sheet designers can increase the weight of their declarations. In the following example:

```
h1 {
    color: black ! important;
    background: white ! important;
}

p {
    font-size: 12pt ! important;
    font-style: italic;
}
```

the first three declarations have increased weight, while the last declaration has normal weight.

The Cascading Algorithm

To find the value for a property of an element, an algorithm must be followed to sort the weights of CSS rules possibly influencing the same property at the same time

1. find all the declarations and their default (inheritance or initial) values
2. sort declarations by importance
3. sort by source: author $>$ reader $>$ browser
4. sort by specificity (see the W3C Standard document for details of the specificity algorithm): more specific $>$ less specific
5. sort by appearance: latter specified $>$ earlier specified

A reader rule with an important declaration will override an author rule with a normal declaration. An author rule with an important declaration will override a reader rule with an important declaration.

An Introduction to Cascading Style Sheets

Laboratory of Computer Technologies L-A

Laboratorio di Tecnologie Informatiche L-A

Andrea Omicini & Giulio Piancastelli
{andrea.omicini, giulio.piancastelli}@unibo.it

Ingegneria Due
Alma Mater Studiorum—Università di Bologna a Cesena

Academic Year 2007/2008