CASCADING STYLE SHEETS

CONCEPT

- Language used to associate **styles** to documents
  - Companion specification to HTML
    - But can be applied to any document structured with a tree (e.g. HTML, XML, SVG)
- Separation CSS / HTML
  - To manage **presentation aspects** (CSS) separately from **structural aspects** (HTML)
  - To present the content differently to different users using different CSS
  - To present different HTML content with the same presentation aspects, same CSS
- Demonstration
  - Deactivate CSS

A BIT OF HISTORY

- CSS 1.0 (1996)
- CSS 1.0 (2nd ed., 1999)
- CSS 2.1 (2011):
  - Stable version, implemented interoperably by browsers
- CSS 3:
  - Modular specification of CSS 2.1
  - Many additions (50+ modules, see list of specifications)
  - Partly implemented by browsers

PRINCIPLES

- Language based on **rules** to be associated with document elements
- Each rule sets some **properties** on some elements
  - A rule is one or more **selectors** and a **declaration block** (block of properties)

- Types of properties (more than 400 defined)
  - Visual properties (`background-*`, `border-*`, ...)
  - Text properties (`text-*`, `font-*`, `color`, ...)
  - Box properties (`padding-*`, `margin-*`, ...)
  - other properties (`visibility`, `display`, `z-index`, ...)

- **Style Sheet**
  - A set of rules in a separate file is a style sheet
  - Multiple style sheets can be applied to a document
    - Author style sheets
    - User style sheets
    - Device Style sheets

DECLARATION OF PROPERTIES

- each property is declared using the syntax: property_name + ':' + value

```
font-weight: 600        /* property with a unitless number value */

font-size: 16px         /* property with a number value with units */

width: 99%              /* property with a percentage value */

background-color: red   /* property with a keyword value */

font-family: 'Arial'    /* property with a string value */

background-image: url('http://my.server.com/clear.png') /* property with a complex
```

- use of ; to group properties applying to the same element(s)

```
background-color: red; font-size: 16px;
color: red;
width: 50%;
```

CSS UNITS

- Size and position units
  - Absolute units
    - px
    - pt, pc, cm, mm, in
      - 1in = 2.54cm = 25.4mm = 72pt = 6pc
  - Relative units
    - percentage units (%)
    - Font-relative units
      - em,ex,ch,rem
    - Viewport relative units
      - vw,vh,vmin,vmax
- Other units
  - deg,grad,rad,turn
  - s,ms
  - Hz,kHz
  - dpi,dpcm,dppx

SELECTORS

- Select to which element(s) a block of properties apply (using { })
  - Selecting elements in the document tree by tag name

```
p { /* these properties apply to all p elements in the page */
  border-style:solid;
  border-width:5px;
}
```

  - Selecting using multiple tag names (separated by a comma)

```
h1, em { /* these properties apply to all h1 and em elements in the page */
  color: blue;
}
```

SELECTORS - MORE

- Addressing of 1 specific element in the document tree by `id` attribute using #

```
<!-- HTML -->
<p id="p1">text 1</p> <!-- each paragraph has a unique id attribute -->
<p id="p2">text 2</p>
```

```
/* CSS */
#p2 { /* this property applies to the element whose id is p2 */
    color: red;
}
#p1 { /* this property applies to the element whose id is p1 */
    color: blue;
}
```

- Addressing of several specific elements by `class` name using `.`

```
<!-- HTML -->
<!-- each paragraph has a class attribute with one or more class values -->
<p class="pType1">text 1</p>
<p class="pType1">text 2</p>
```

```
/* CSS */
.pType1 { /* this property applies to all elements whose class attribute contains p
    color: blue;
}
```

LINKING CSS CONTENT WITH HTML CONTENT

- Via the `style` attribute (**inline stylesheet**)
  - Styles attached to a given element (*syntax without selector*)

```
<p style="color:red;">text</p>
```

  - should be avoided

- Via the `style` element (**internal stylesheet**)
  - Styles attached to a given document

```
<head>
 <style>
 p { color: red; }
 </style>
</head>
```

  - should be avoided

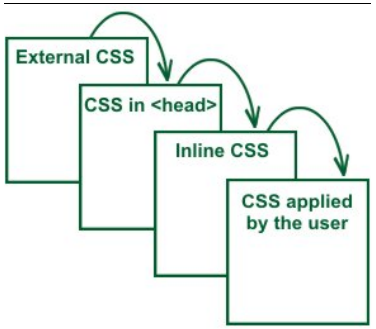
- Via an external stylesheet (separate file)
  - Styles can be attached to a document

```
<link href="file.css" type="text/css" rel="stylesheet"/>
```

- should be preferred

CSS CASCADE

- If different rules conflict (e.g. when multiple style sheets are used)
- The rule that has precedence is determined by:
  - media type of style sheet
  - origin of rule (user agent, user, author, !important author, !important user)
  - specificity of the selector
  - order in file



EXAMPLE OF A CSS PROPERTY DEFINITION

■ The **border-top-width** property

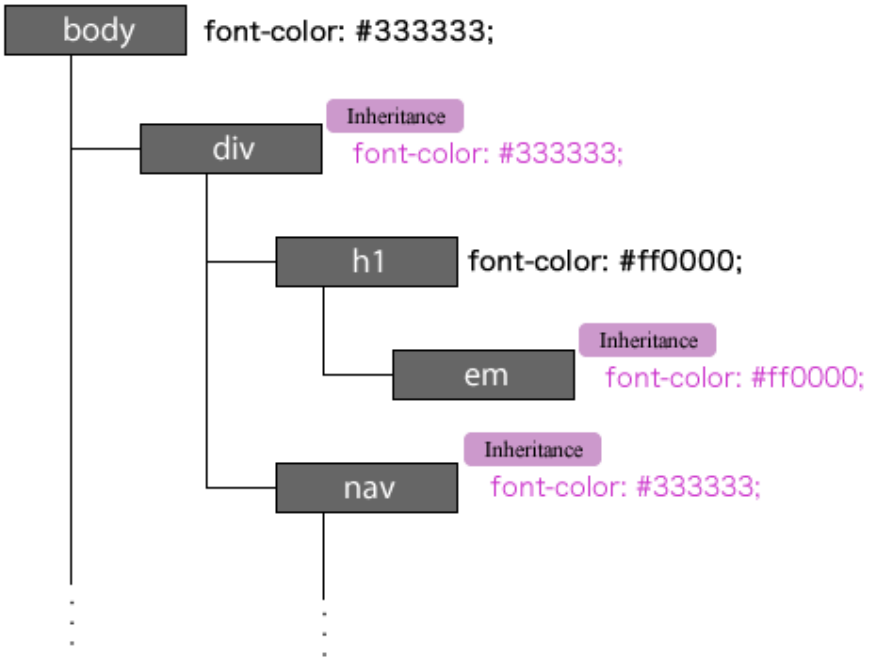
Syntax:

<length> | thin | medium | thick

Definition:

Initial value	`medium`
Applies to	all elements. It also applies to `::first-letter`.
Inherited	no
Media	visual
Computed value	the absolute length or `0` if `border-top-style` is `none` or `hidden`
Animatable	yes, as a length

CSS INHERITANCE



CSS INHERITANCE

■ For a given element, if the value for a given property is **not specified**, the value is obtained as follows:

- if the property is inheritable by default (i.e. "inherited: yes"),
  - if the element has a parent in the DOM tree, the **computed value** on that parent is use

```
p { color: green }
```

```
<p>The text and the span will be <span>green</span> because 'color' is inheritable.
```

  - otherwise (for the root), the **initial value** is used.

- if not (i.e. "inherited: no"), the **initial value** is used

```
p { border-width: 1px }
```

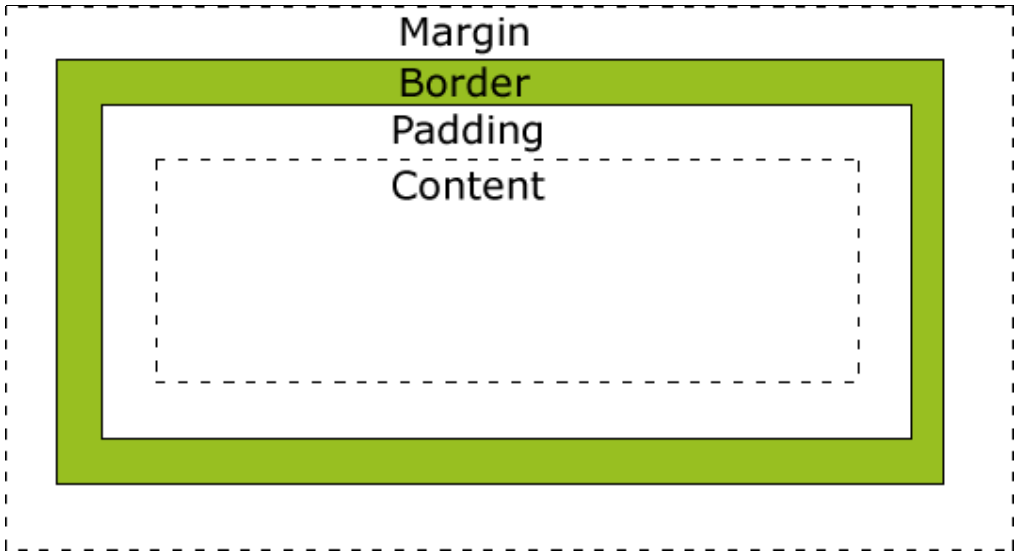
```
<p>Only the text will have <span>a border</span> because 'border-width' is not inhe
```

■ The computed value is obtained:

- by converting a relative value (when possible) to an absolute value
- otherwise (% values when layout is involved), using the relative value

THE CSS BOX MODEL

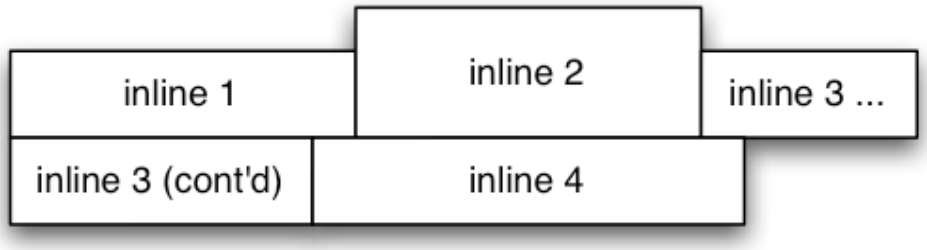
- Each element in the DOM produces zero, one or several boxes depending on the type of element
  - The page rendering consists in displaying those boxes
- Each box has generic properties that controls some generic aspects: margin, border, padding

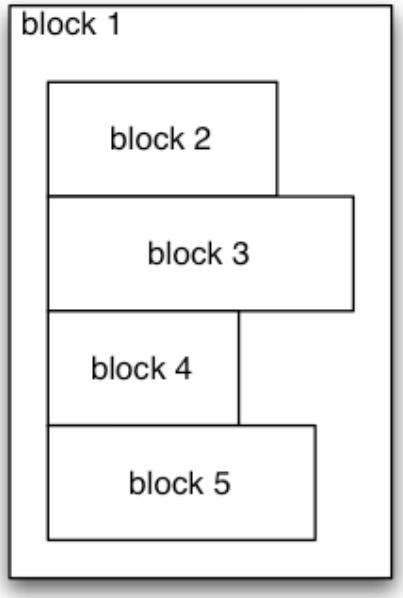


- The layout (size and position) of a box depends on multiple factors:
  - The size of the box and of its content (e.g. images)
  - The type of box (block, inline, ...)
  - The positioning scheme: normal, absolute, float
  - The other elements and boxes around (siblings, parent, containers)
  - The viewport (e.g. the window size)

CSS BOX TYPES

- There are 2 main types of boxes:
  - **block** boxes: Boxes that don't display on the same line as the previous box and as the next box
    - Sizing properties such as `width` and `height` can be used.
  - **inline** boxes: Boxes that stay on the same line as the previous box and the next box (when possible)





- The type of box is defined by the standard:
  - block boxes: p, div, h1, h2, footer ...
  - inline boxes: a, img, span ...
- The default type can be overriden by the **display** property

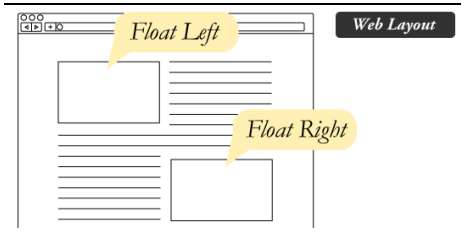
```
<p>A first par</p>
<p>A second par</p>
<a>A first link</a>
<a>A second link</a>
```

```
p { display: inline; }
a { display: block; }
```

CSS POSITIONING SCHEMES

- CSS defines the `position` property with the values
  - `static`: default value
  - `relative`: moved compared to its original position (initial place left empty)
  - `absolute`: positioned relative to the origin of the parent box
  - `fixed`: positioned relative to the window

- Floats



- z-index

RESPONSIVE DESIGN

- Principles
  - Design pages that adapt to the screen size using CSS Media Queries

---

CSS MEDIA QUERIES

- Adapt the CSS rules to apply based on client characteristics
  - Screen size, aspect-ratio, resolution or orientation
  - Type of device (pc, mobile, printer ...)
  - Number of colors

```
<link rel="stylesheet" media="screen and (max-width: 1280px)" href="file.css" />
```

or

```
<link rel="stylesheet" href="file-with-mediaqueries.css" />
```

```
@media screen and (max-width: 1280px)
{
    /* SomeCSS ruleshere */
}
```

ADVANCED SELECTORS

All elements:

```
* { }
```

Elements with a given attribute:

```
element[foo] {}
```

Elements with a given attribute value:

```
element[foo='bar'] {}
```

Element as a descendant of another:

```
h3 em { }
```

Element as a child of another:

```
div > p { }
```

Element preceded by another:

```
p ~ div { }
```

Pseudo-classes

```
a:link {color:#FF0000;}
a:visited {color:#00FF00;}
a:hover {color:#FF00FF;}
a:active {color:#0000FF;}
li:nth-child(2) {color:#0000FF;}
```

Pseudo-elements

```
:first-line { color: red; }
```

ADVANCED PROPERTY NOTATION

- Short-hand notation
  - group several related properties into one
  - specific order without missing properties

```
padding: 4px 9px;
border: 1px solid #fff;
box-shadow: inset 0 1px 2px rgba(0,0,0,.3);
```

- Vendor-prefix notation (`-o-`, `-ms-`, `-moz-`, `-webkit-`,...)

```
-moz-box-shadow: inset 0 1px 2px rgba(0,0,0,.3);
-webkit-box-shadow: inset 0 1px 2px rgba(0,0,0,.3);
```

AUTHORING CSS

- Many web sites offer free CSS templates
  - http://www.free-css.com/
  - http://templated.co/
  - ...
- CSS tools
  - Pre-processors to generate CSS
    - SASS
    - LESS
  - WYSIWYG editors
    - BlueGriffon
    - SelfCSS
  - Responsive front-end frameworks
    - Bootstrap
    - Foundation