Cascading Style Sheet

Fajiang Yu, fjyu@whu.edu.cn

School of Computer, Wuhan University

2017.3

Agenda

1. HTML and Style Sheet
2. CSS Introduction
3. CSS Selectors
4. CSS Box Model
5. CSS Visual Formatting Model
6. CSS generated content, automatic numbering, and lists
7. CSS colors and backgrounds
8. CSS Fonts and Text
9. CSS Table and Multicolumns
10. CSS user interface
11. CSS3 2D/3D Transforms
12. CSS3 Transition
13. CSS3 Animation

Agenda

Agenda

Introduction to style sheet

HTML's stylistic limitations

Style sheets make it easy to

- specify the amount of white space between text lines,
- the amount lines are indented,
- the colors used for the text and the backgrounds
- .....

Introduction to style sheet

HTML 4 provides support for the following style sheet features

- Flexible placement of style information.
  Placing style sheets in separate files makes them easy to reuse.
- Independence from specific style sheet languages
  Example: CSS (Cascading Style Sheets) language
- Cascading
- Media dependencies
- Alternate styles
- Performance concerns

Agenda

Inline style information

`style` attribute specifies style information for the current element.

CSS example

```
<p style="font-size: 12pt; color: fuchsia">style sheet</p>
```

Header style information

The `style` element allows authors to put style sheet rules in the head of the document.
HTML permits any number of `style` elements in the `head` section of a document.

some attributes

- `type`: specifies the style sheet language of the element's contents and overrides the default style sheet language
- `media`: specifies the intended destination medium for style information, such as `projection` and `print`. The default value is `screen`. (Continuous media or page media)

Header style information

CSS `style` declaration example

```
<head>
   <style type="text/css">
    h1 {
       border-width: 1;
       border: solid;
       text-align: center }
  </style>
</head>
```

External style sheets

Authors may separate style sheets from HTML documents.
This offers several benefits:

- Authors and Web site managers may share style sheets across a number of documents (and sites).
- Authors may change the style sheet without requiring modifications to the document.
- User agents may load style sheets selectively (based on media descriptions).

Specifying external style sheets

With the following attributes of the `link` element

- `href`, specifies the location of the style sheet file, a URI
- `type`, indicates the language of the linked (style sheet) resource
- Specify that the style sheet is persistent, preferred, or alternate
  - To make a style sheet persistent, set the `rel` attribute to `stylesheet` and don't set the `title` attribute
  - To make a style sheet preferred, set the `rel` attribute to `stylesheet` and name the style sheet with the `title` attribute
  - To specify an alternate style sheet, set the `rel` attribute to `alternate stylesheet` and name the style sheet with the `title` attribute

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

Agenda

A brief history of CSS

CSS 3

CSS3 has been split into “modules”. It contains the “old CSS specification” (which has been split into smaller pieces). In addition, new modules are added.

The CSS3 specification is still under development by W3C. However, many of the new CSS3 properties have been implemented in modern browsers.

Some of the most important CSS3 modules are:

- Selectors, Box Model, Backgrounds and Borders
- Image Values and Replaced Content
- Text Effects, 2D/3D Transformations
- Animations, Multiple Column Layout, User Interface

CSS Level 2 Revision 1

CSS Level 1

A brief CSS 2.1 tutorial for HTML

To set the text color of the `h1` elements to red, you can write the following CSS rules:

```
h1 { color: red }
```

A CSS rule consists of two main parts

- selector `h1` and
- declaration `color: red`
  The declaration has two parts
  - property name `color` and
  - property value `red`

CSS 2.1 has more than 90 properties.

Rule sets, declaration blocks, and selectors

A rule set (also called ”rule”) consists of a selector followed by a declaration block.

A declaration block starts with a left curly brace and ends with the matching right curly brace. In between there must be a list of zero or more semicolon-separated declarations. (The last declaration may also be followed by a semicolon.)

The selector consists of everything up to (but not including) the first left curly brace.

Declarations and properties

A declaration is either empty or consists of a property name, followed by a colon, followed by a property value.
Around each of these there may be white space.

Multiple declarations for the same selector may be organized into semicolon separated groups.

The following rules

```
h1 { font-weight: bold }
h1 { font-size: 12px }
```

are equivalent to

```
h1 {
    font-weight: bold;
    font-size: 12px; }
```

Every CSS property has its own syntactic and semantic restrictions on the values it accepts.

Shorthand properties

They allow authors to specify the values of several properties with a single property.

Example of multiple style rules

```
h1 {
   font-weight: bold;
   font-size: 12pt;
   line-height: 14pt;
   font-family: Helvetica;
   font-variant: normal;
   font-style: normal; }
```

may be rewritten with a single shorthand property

```
h1 { font: bold 12pt/14pt Helvetica }
```

When values are omitted from a shorthand form, each "missing" property is assigned its initial value.

Comments

Comments begin with the characters `/*` and end with the characters `*/`.

Values

Integers and real numbers

Lengths

- Relative units
  - `em`: the 'font-size' of the relevant font
  - `ex`: the 'x-height' of the relevant font
- Absolute length units
  - `in`: inches - `1in` is equal to `2.54cm`
  - `cm`: centimeters
  - `mm`: millimeters
  - `pt`: points - the points used by CSS are equal to 1/72nd of `1in`
  - `pc`: picas - `1pc` is equal to `12pt`
  - `px`: pixel units - `1px` is equal to `0.75pt`

Values

Percentages

URLs and URIs

Counters

Example(s)

```
p { counter-increment: par-num }
h1 { counter-reset: par-num }
p:before { content: counter(par-num, upper-roman) ". " }
```

Values

Colors: either a keyword or a numerical RGB specification

Example(s)

```
body { color: black; background: white }
em { color: #f00 } /* #rgb */
em { color: #ff0000 } /* #rrggbb */
em { color: rgb(255,0,0) }
em { color: rgb(100%, 0%, 0%) }
```

Values

Strings: can either be written with double quotes or with single quotes.

initial value

`inherit`

`auto`

Agenda

Type selectors

A type selector matches the name of a document language element type.

Example(s)

```
h1 { font-family: sans-serif }
```

Universal selector

The universal selector, written `*`, matches the name of any element type.

If the universal selector is not the only component of a simple selector (universal and type selectors), the `*` may be omitted.
For example

- `*[lang=fr]` and `[lang=fr]` are equivalent
- `*.warning` and `.warning` are equivalent
- `*#myid` and `#myid` are equivalent

Attribute selectors

To specify rules that match elements which have certain attributes. Attribute selectors may match in four ways

- `[att]`
- `[att=val]`
- `[att~=val]`
  value is a white space-separated list of words, one of which is exactly `val`
- `[att|=val]`
  value either being exactly `val` or beginning with `val` immediately followed by `-`

Example(s)

```
h1[title] { color: blue; }
```

Class selectors

Working with **HTML**, authors may use the period (`.`) notation as an alternative to the `~=` notation when representing the `class` attribute. The attribute value must immediately follow the period.

Example(s)

```
*.pastoral { color: green }
.pastoral { color: green }
h1.pastoral { color: green }
h1[class~=pastoral] { color: green }
```

ID selectors

CSS ID selectors match an element instance based on its identifier.
A CSS ID selector contains a # immediately followed by the ID value.

Example(s)

```
h1#chapter1 { text-align: center }
*#z98y { letter-spacing: 0.3em }
```

Pseudo-classes

`:first-child` pseudo-class

Example(s)

```
div > p:first-child { text-indent: 0 }
p:first-child em { font-weight: bold }
```

Pseudo-classes

The link pseudo-classes
In HTML4, the link pseudo-classes apply to `a` elements with an `href` attribute.

- The `:link` pseudo-class applies for links that have not yet been visited.
- The `:visited` pseudo-class applies once the link has been visited by the user.

Example(s)

```
a:link { color: red }
:link { color: red }
a.external:visited { color: blue }
```

Pseudo-classes

The dynamic pseudo-classes

- `:hover`
- `:active`
- `:focus`

Example(s)

```
a:hover { color: yellow } /* user hovers */
a:active { color: lime } /* active links */
a:focus { background: yellow }
```

Pseudo-classes

The language pseudo-class `:lang`

Pseudo-elements

Example(s)

```
p:first-line { text-transform: uppercase }
p:first-letter { font-size: 3em; font-weight: normal }

h1:before { content: counter(chapno, upper-roman) ". " }
h1:after { content: counter(chapno, upper-roman) ". " }
```

Selector Grouping

When several selectors share the same declarations, they may be grouped into a comma-separated list.

Example(s)

```
h1 { font-family: sans-serif }
h2 { font-family: sans-serif }
h3 { font-family: sans-serif }
```

is equivalent to

```
h1, h2, h3 { font-family: sans-serif }
```

Descendant selectors

A descendant selector is made up of two or more selectors separated by white space.
A descendant selector of the form `a b` matches when an element `b` is an arbitrary descendant of some ancestor element `a`.

Example(s)

```
h1 em { color: blue }
```

Child selectors

A child selector matches when an element is the child of some element.
A child selector is made up of two or more selectors separated by >.

Example(s)

```
body > p { line-height: 1.3 }
```

Adjacent sibling selectors

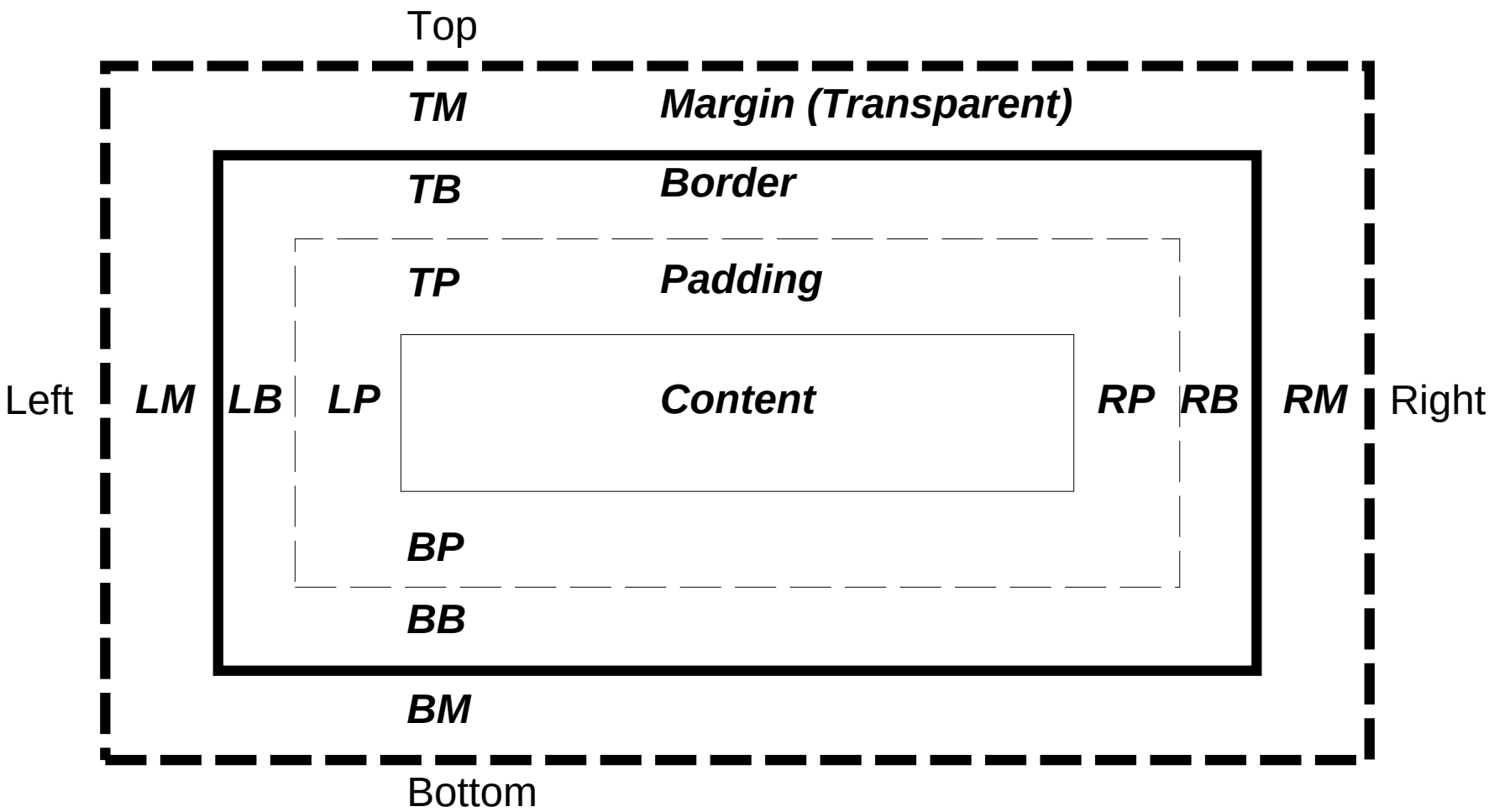
Adjacent sibling selectors have the following syntax: `E1 + E2`. The selector matches if `E1` and `E2` share the same parent in the document tree and `E1` immediately precedes `E2`.

Example(s)

```
h1 + h2 { margin-top: -5mm }
```

Agenda

Box Dimensions



Box Dimensions

Example(s)

```
<style type="text/css">
   ul {
      background: yellow;
      margin: 12px 12px 12px 12px;
      padding: 3px 3px 3px 3px; }
   li {
      color: white; /* text color is white */
      background: blue; /* Content, padding will be blue */
      margin: 12px 12px 12px 12px;
      padding: 12px 0px 12px 12px; /* Note 0px padding right */
      list-style: none /* no glyphs before a list item */ }
   li.withborder {
      border-style: dashed;
      border-width: medium; /* sets border width on all sides */
      border-color: lime; }
</style>
```

Margin properties and padding properties

Margin properties

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

- `margin`

Padding Properties

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

- `padding`

Border properties

Border width

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

- `border-width`

Some values

- `thin`
- `medium`
- `thick`

Border color

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

- `border-color`

Border properties

Border style

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

- `border-style`

Border style values

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

Border properties

Border shorthand properties

- `border-top`
- `border-right`
- `border-bottom`
- `border-left`

- `border`

Example(s)

```
p { border: solid red }
p {
    border-top: solid red;
    border-right: solid red;
    border-bottom: solid red;
    border-left: solid red }
```

CSS3 Borders

New Properties

- `border-radius`
- `box-shadow`
- `border-image`

Example(s)

```
div {
    border: 2px solid;
    border-radius: 25px;
    box-shadow: 10px 10px 5px #888888; }
```

Agenda

Block-level elements and block boxes

Block-level elements are specified by the following values of `display` property

- `block`
- `list-item`
- `table`

Each block-level element generates a **principal** block-level box that contains (Except for `table` boxes, and **replaced** elements, such as an image, embedded document, or applet.)

- descendant boxes and
- generated content.

Some block-level elements may generate **additional** boxes in addition to the principal box: `list-item` elements.

Inline-level elements and inline boxes

Inline-level elements are specified by the following values of `display` property

- `inline`
- `inline-table`
- `inline-block`

Inline-level elements generate **inline-level boxes**, not **inline boxes**.

HTML grouping elements

The `div` element, define content to be block-level
The `span` element, define content to be inline

in conjunction with the `id` and `class` attributes,
offer a generic mechanism for adding structure to documents.

The display property

Values

- `block`, `list-item`
- `inline`, `inline-block`
- `none`
  Causes an element and its contents (its descendant elements and responding contents) to not appear in the formatting structure (do not generate any boxes).
- `table`, `inline-table`, `table-row-group`, `table-column`, `table-column-group`, `table-header-group`, `table-footer-group`, `table-row`, `table-cell`, `table-caption`
  cause an element to behave like a table element

Positioning schemes

Three positioning schemes

- Normal flow
  - block formatting, boxes are laid out one after the other, vertically, beginning at the top of a containing block
  - inline formatting, boxes are laid out horizontally, one after the other, beginning at the top of a containing block
  - relative positioning, once a box has been laid out according to the normal flow or floated, it may be shifted relative to this position
- Floats, a float is a box that is shifted to the left or right on the current line
- Absolute positioning
  - a box is explicitly offset with respect to its containing block
  - Fixed positioning, the containing block is established by the **viewport**

The `position` property and box offsets

The `position` property values

- `static`
- `relative`
- `absolute`
- `fixed`

Box offsets properties

- `top`
- `right`
- `bottom`
- `left`

The `float` and `clear` property

`float` values

- `left`
- `right`
- `none`

`clear` values

- `left`
- `right`
- `both`
- `none`

Positioning schemes examples

```
<body>
   <p>Beginning of body contents.
      <span id="outer"> Start of outer contents.
         <span id="inner"> Inner contents. </span>
         End of outer contents.
      </span>
      End of body contents.
   </p>
</body>
```

Positioning schemes examples

```
body {
    display: block;
    font-size:12px;
    line-height: 200%;
    width: 400px;
    height: 400px }
p { display: block }
span { display: inline }
```

Positioning schemes examples

Normal flow

```
#outer { color: red }
#inner { color: blue }
```

Positioning schemes examples

Relative positioning

```
#outer {
    position: relative;
    top: -12px;
    color: red }
#inner {
    position: relative;
    top: 12px;
    color: blue }
```

Positioning schemes examples

Floating a box

```
#outer { color: red }
#inner {
    float: right;
    width: 130px;
    color: blue }
```

Positioning schemes examples

Floating a box

```
<body>
   <p>Beginning of body contents.
      <span id="outer">Start of outer contents.
         <span id="inner">Inner contents.</span>
         <span id="sibling">Sibling contents.</span>
         End of outer contents.
      </span>
      End of body contents.
   </p>
</body>
```

Positioning schemes examples

Floating a box

```
#inner {
    float: right;
    width: 130px;
    color: blue }
#sibling { color: red }
```

```
#inner {
    float: right;
    width: 130px;
    color: blue }
#sibling {
    clear: right;
    color: red }
```

Positioning schemes examples

Absolute positioning

```
#outer {
    position: absolute;
    top: 200px;
    left: 200px;
    width: 200px;
    color: red; }
#inner { color: blue }
```

Positioning schemes examples

Absolute positioning

```
#outer {
   position: relative;
   color: red }
#inner {
   position: absolute;
   top: 200px;
   left: -100px;
   height: 130px;
   width: 130px;
   color: blue; }
```

```
#outer { color: red }
#inner {
   position: absolute;
   top: 200px;
   left: -100px;
   height: 130px;
   width: 130px;
   color: blue; }
```

If the element has `position: absolute`, the containing block is established by the nearest ancestor with a `position` of `absolute`, `relative` or `fixed`.

Initial containing block

Layered presentation

The `z-index` property

Text direction

The `direction` and `unicode-bidi` properties

Values

- `ltr`
- `rtl`

Content width

The `width` property

The `min-width` and `max-width` properties

Content height

The `height` property

The `min-height` and `max-height` properties

Line height and vertical align

`line-height` property values

- `normal`
- *number*
- *length*
- *percentage*

`vertical-align` property values

- `baseline`
- `sub`
- `super`
- `top`
- `text-top`
- `middle`
- `bottom`
- `text-bottom`

The `overflow` property

Values

- `visible`
- `hidden`
- `scroll`

The clip property

Values

- rect(<top>, <right>, <bottom>, <left>)

The `visibility` property

Values

- `visible`
- `hidden`
- `collapse`

Agenda

The content property

The `content` property is used with the `:before` and `:after` pseudo-elements to generate content in a document.

Some values

- `none`
- `normal`
- *string*
- *uri*
- *counter*
- `attr(X)`

```
h1:before {
    content: "Chapter: ";
    display: inline;
}
```

The quotes property

This property specifies quotation marks for any number of embedded quotations.

Some values

- `none`
- [*string string*] +
  for the `open-quote` and `close-quote` values of the `content` property are taken from this list of pairs of quotation marks

```
q { quotes: ’ ”’ ’ ”’ ”’ ” ”’ ” }

q:before { content: open-quote }
q:after { content: close-quote }

<p><q>Quote from <q>Textbook</q></q></p>
```

Automatic counters

The `counter-reset` property

Some values

- `none`
- [*identifier integer*?] +
  contains a list of one or more names of counters, each one optionally followed by an integer, the default is 0

Automatic counters

The `counter-increment` property

Some values

- `none`
- [*identifier integer*?] +
  accepts one or more names of counters (identifiers), each one optionally followed by an integer, the default increment is 1

Automatic counters

The `content` property values about **counter**

- `counter(name)` or `counter(name, style)`
- `counters(name, string)` or `counters(name, string, style)` generates a string composed of all of the counters with the same name that are in scope, separated by a given string

Counter style type:

`disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | none`

Automatic counters

Example(s)

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

h1:before {
    content: "Chapter " counter(chapter) ". ";
    counter-increment: chapter; }
h1 { counter-reset: section; }

h2:before {
    content: counter(chapter) "." counter(section) " ";
    counter-increment: section; }
```

Automatic counters

Example(s)

```
ol { counter-reset: item }
li { display: block }
li:before {
   content: counters(item, ".") " ";
   counter-increment: item }
```

Lists

The `list-style-type` property

Values

`disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | none`

Lists

The `list-style-image` property

Values
uri

The `list-style-position` property

Values

- `inside`
- `outside`

Lists

The `list-style` property is a shorthand notation for setting the three properties `list-style-type`, `list-style-image`, and `list-style-position`.

```
ul { list-style: upper-roman inside }
```

Agenda

Foreground color

The `color` property

The background

The `background-color` property

The background

The `background-image` property

The `background-repeat` property
Values
`repeat | repeat-x | repeat-y | no-repeat`

The `background-attachment` property
Values
`scroll | fixed`

The `background-position` property
Values
percentage, *length*,
`top`, `right`, `bottom`, `left`, `center`

The background

The `background` property is a shorthand property for setting the individual background properties (i.e., `background-color`, `background-image`, `background-repeat`, `background-attachment` and `background-position`).

```
p { background: url("chess.png") gray 50% repeat fixed }
```

The background

CSS3 Multiple Background Images

```
background:url(img_tree.gif),url(img_flwr.gif);
```

CSS3 Background Properties

- `background-clip`
- `background-origin: content-box | padding-box | border-box`
- `background-size`

The background

CSS3 defines two types of gradients:

- Linear Gradients (goes down/up/left/right/diagonally)
- Radial Gradients (defined by their center)

```
background: linear-gradient(direction, color-stop1,  color-stop2, ...);

background: linear-gradient(red, blue);
background: linear-gradient(to right, red , blue);
background: linear-gradient(to bottom right, red , blue);
background: linear-gradient(180deg, red, blue);
background: linear-gradient(red, green, blue);
background: linear-gradient(to right, red,orange,yellow,green,blue,
                                      indigo,violet);
background: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));
background: repeating-linear-gradient(red, yellow 10%, green 20%);
```

The background

CSS3 Radial Gradients:

```
background: radial-gradient(center, shape size, start-color, ..., last-color);

background: radial-gradient(red, green, blue);
background: radial-gradient(red 5%, green 15%, blue 60%);
background: radial-gradient(circle | ellipse, red, yellow, green);
background: radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
/* farthest-side, closest-corner, farthest-corner */
background: repeating-radial-gradient(red, yellow 10%, green 15%);
```

Agenda

The font-family property

Values

$[[\ \textit{family-name} \mid \textit{generic-family}\ ][,\ \textit{family-name} \mid \textit{generic-family}]*\ ]$

```
body { font-family: Gill, Helvetica, sans-serif }
```

Generic font families

- serif (e.g., Times)
- sans-serif (e.g., Helvetica)
- cursive (e.g., Zapf-Chancery)
- fantasy (e.g., Western)
- monospace (e.g., Courier)

CSS Fonts

The `font-style` property
Values
`normal | italic | oblique`

The `font-variant` property
Values
`normal | small-caps`

The `font-weight` property
Values
`normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900`

The `font-size` property

Values

- *absolute-size*
  `xx-small | x-small | small | medium | large | x-large | xx-large`
- *relative-size*
  `larger | smaller`
- *length*
- *percentage*

The font property

The font property is a shorthand property for setting font-style, font-variant, font-weight, font-size, line-height and font-family.

CSS3 Web Fonts

CSS3 Font Descriptors used in @font-face rule:

- font-family
- src
- font-stretch
- font-style
- font-weight
- unicode-range

```
@font-face {
 font-family: myFirstFont;
 src: url(sansation_light.woff); }

div {
 font-family:myFirstFont; }
```

Text

The `text-indent` property
Values
length | *percentage*

The `text-align` property
Values
`left | right | center | justify`

The `text-decoration` property
Values
`none | [ underline || overline || line-through || blink ]`

Text

The `letter-spacing` and `word-spacing` properties
Values

- `normal`
- *length*

The `text-transform` property
Values
`capitalize | uppercase | lowercase | none`

The `white-space` property
Values
`normal | pre | nowrap | pre-wrap | pre-line`

Text

CSS3 Text Properties

- `hanging-punctuation, punctuation-trim`
- `text-align-last, text-emphasis, text-justify, text-outline, text-overflow`
- `text-shadow, text-wrap, word-break, word-wrap`

```
h1 { text-shadow: 5px 5px 5px #FF0000; }
p { word-wrap:break-word; }
```

Agenda

CSS Table Model

Using `display` property values to assign table formatting rules to an arbitrary element.

The default style sheet for HTML 4 elements

```
table { display: table }
tr { display: table-row }
thead { display: table-header-group }
tbody { display: table-row-group }
tfoot { display: table-footer-group }
col { display: table-column }
colgroup { display: table-column-group }
td, th { display: table-cell }
caption { display: table-caption }
```

Columns

The following properties apply to column and column-group elements

- `border`
- `background`
- `width`
- `visibility`

Caption position and alignment

The `caption-side` property

Values
`top | bottom`

Table width algorithms

The `table-layout` property

Values

- `auto`
  The table's width is given by the width of its columns (and intervening borders).
- `fixed`
  The horizontal layout of the table does not depend on the contents of the cells; it only depends on the table's width, the width of the columns, and borders or cell spacing.

Borders

The `border-collapse` property selects a table's border model

- `collapse`
- `separate`

Borders

The separated borders model

The `border-spacing` property
Value
length length?
The lengths specify the distance that separates adjoining cell borders.
If one length is specified, it gives both the horizontal and vertical spacing.
If two are specified, the first gives the horizontal spacing and the second the vertical spacing.

The `empty-cells` property
Values
`show | hide`

Borders

The collapsing border model

In the collapsing border model, it is possible to specify borders that surround all or part of a cell, row, row group, column, and column group.

```
table {
   border-collapse: collapse;
   border: 5px solid yellow; }
*#col1 { border: 3px solid black; }
td {
   border: 1px solid red;
   padding: 1em; }
td.cell5 { border: 5px dashed blue; }
td.cell6 { border: 5px solid green; }
```

Borders

The collapsing border model

```
<table>
   <col id="col1" /><col id="col2" /><col id="col3" />
   <tr id="row1">
      <td> 1 </td>
      <td> 2 </td>
      <td> 3 </td>
   </tr>
   <tr id="row2">
      <td> 4 </td>
      <td class="cell5"> 5 </td>
      <td class="cell6"> 6 </td>
   </tr>
   <tr id="row3">
      <td> 7 </td>
      <td> 8 </td>
      <td> 9 </td>
   </tr>
</table>
```

Borders

Some of the values of the `border-style` have different meanings in tables than for other elements.

`none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset`

CSS3 Multiple Columns

Properties

- `column-count, column-span`
- `column-fill, column-gap, column-width`
- `column-rule, column-rule-color`
- `column-rule-style, column-rule-width`
- `columns`

Example

```
div {
  column-count:3;
  column-gap:40px;
  column-rule:3px outset #ff00ff; }
```

Agenda

The cursor property

Values

```
[ [<uri> ,]* [ auto | crosshair | default | pointer | move |
e-resize | ne-resize | nw-resize | n-resize | se-resize |
sw-resize | s-resize | w-resize | text | wait | help |
progress ] ]
```

Dynamic outlines

The properties

- `outline-color`
- `outline-style`
- `outline-width`

The `outline` property is a shorthand property.

Example(s)

```
:focus { outline: thick solid black }
:active { outline: thick solid red }
```

CSS3 User-interface

Properties

- `appearance`
- `box-sizing`
- `icon`
- `nav-down, nav-index, nav-left, nav-right, nav-up`
- `outline-offset`
- `resize`

Example(s)

```
div {
  resize:both;
  overflow:auto; }
```

Agenda

CSS3 2D Transforms

CSS3 Transform Properties

- `transform`
- `transform-origin`

CSS3 Transform Methods

- `matrix(n,n,n,n,n,n)`
- `translate(x,y)`, `translateX(n)`, `translateY(n)`
- `scale(x,y)`, `scaleX(n)`, `scaleY(n)`
- `rotate(angle)`
- `skew(x-angle,y-angle)`, `skewX(angle)`, `skewY(angle)`

CSS3 2D Transforms

Example(s)

```
transform: translate(50px,100px);
transform: rotate(30deg);
transform: scale(2,4);
transform: skew(30deg,20deg);
transform: matrix(0.866,0.5,-0.5,0.866,0,0);
```

CSS3 3D Transforms

Agenda

CSS3 Transition

CSS3 Transition Properties

- `transition`
- `transition-property`, `transition-duration`
- `transition-timing-function`, `transition-delay`

Example(s)

```
div:hover { width:300px; }

div { transition: width 2s, height 2s, transform 2s; }

transition-property: width;
transition-duration: 1s;
transition-timing-function: linear;
transition-delay: 2s;
```

Agenda

CSS3 Animation

Properties

- `@keyframes`
- `animation`
- `animation-name, animation-duration, animation-timing-function`
- `animation-delay, animation-iteration-count`
- `animation-direction, animation-play-state`

CSS3 Animation

Example(s)

```
@keyframes myfirst {
  0%   { background: red; left:0px; top:0px; }
  25%  { background: yellow; left:200px; top:0px; }
  50%  { background: blue; left:200px; top:200px; }
  75%  { background: green; left:0px; top:200px; }
  100% { background: red; left:0px; top:0px; } }

div { animation: myfirst 5s linear 2s infinite alternate; }
```

Thank You!
Any Questions?