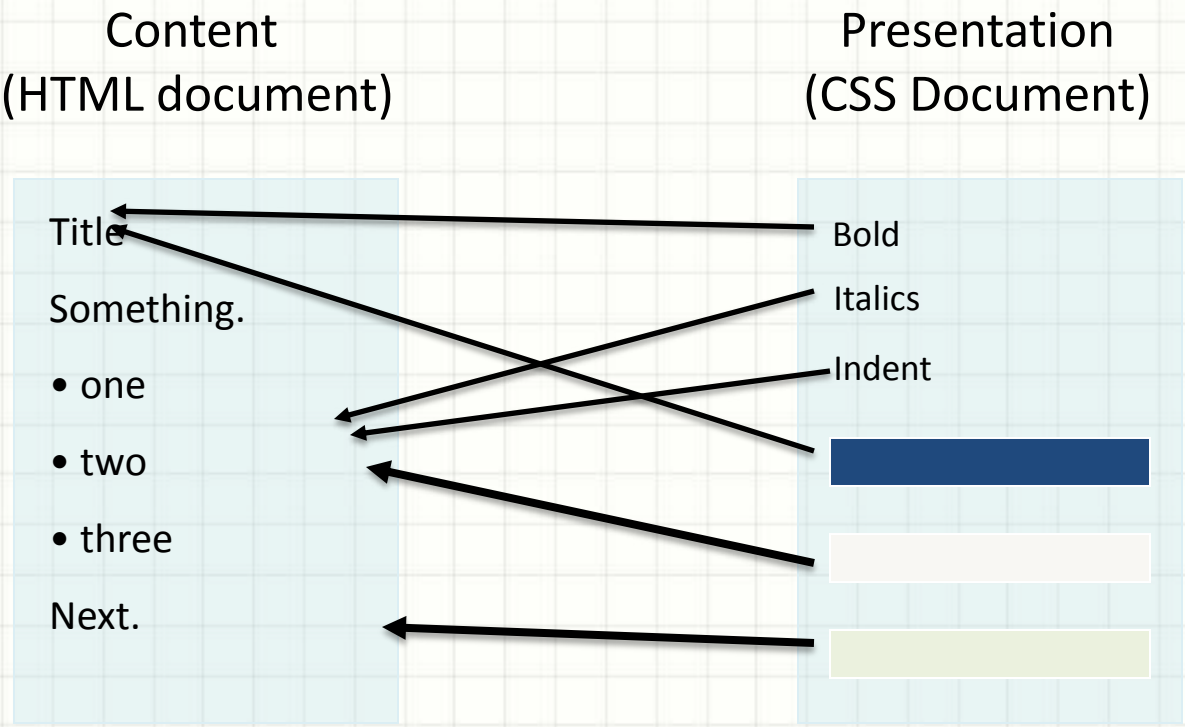
CSS – BASICS

Cascading Stytlesheets

Table of Contents

CSS: A New Philosophy

- Separate content from presentation!



The Resulting Page

Title

Something.

- *One*
- *Two*
- *Three*

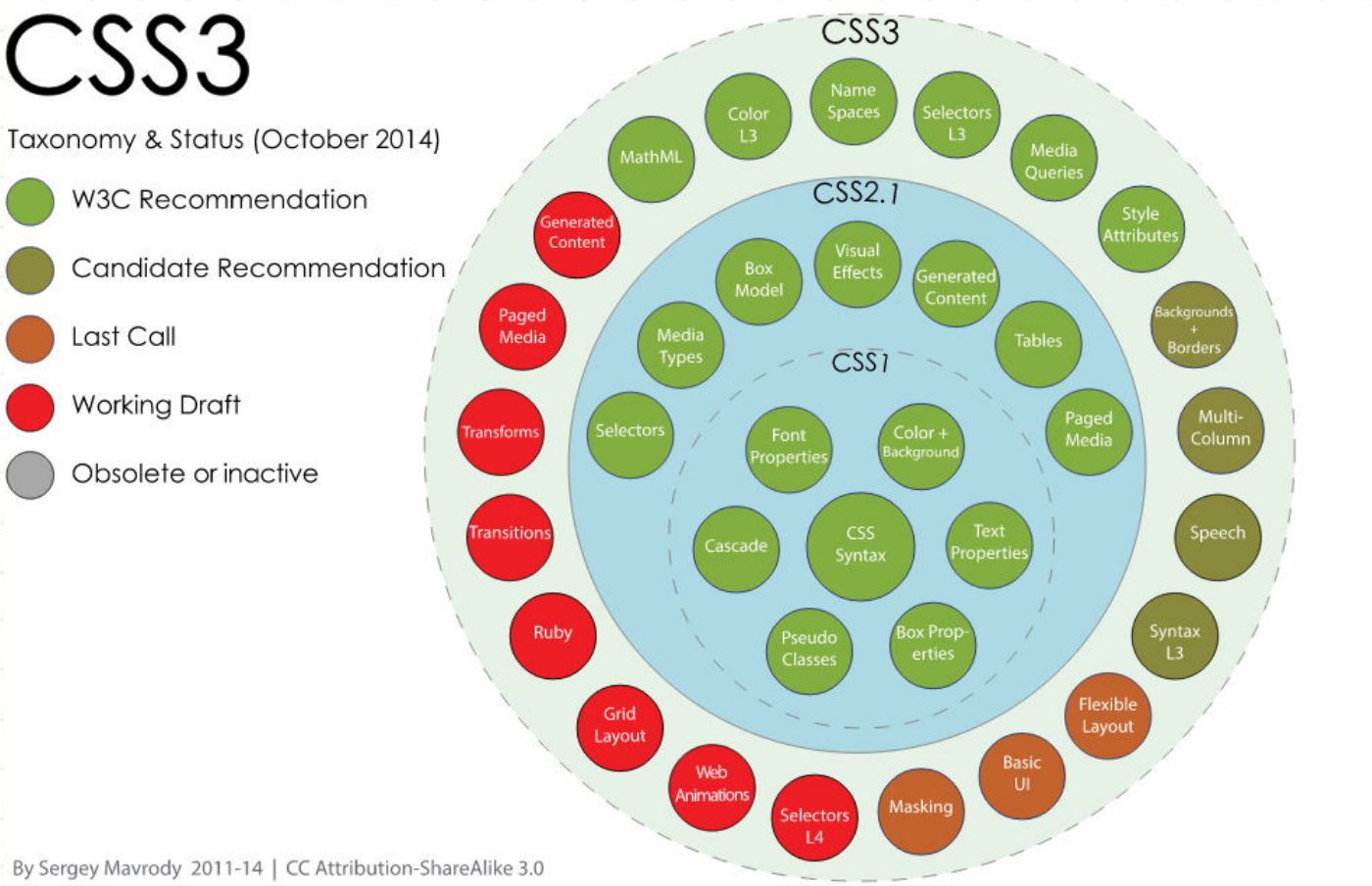
Next.

CSS Introduction

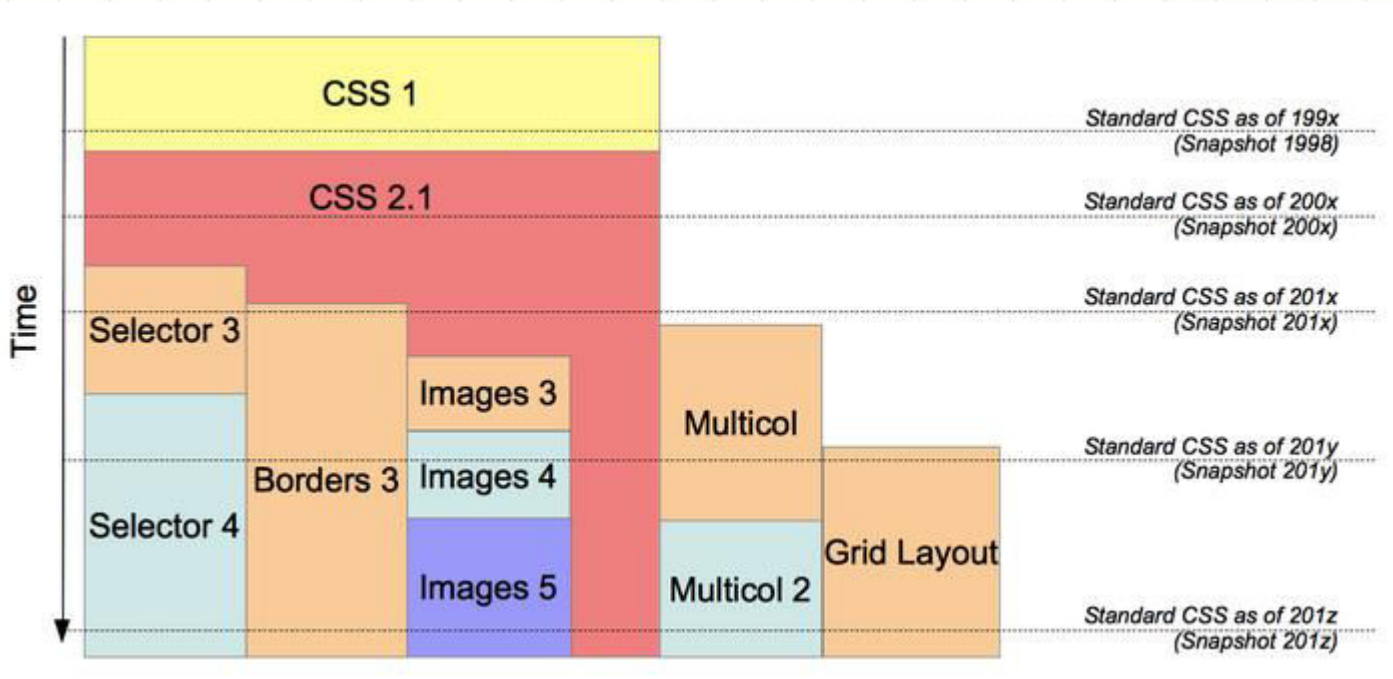
- Cascading Style Sheets (CSS)
  - Used to describe the presentation of documents
  - Define sizes, spacing, fonts, colors, layout, etc.
  - Improve content accessibility
  - Improve flexibility
- Designed to separate presentation from content
- Due to CSS, all HTML presentation tags and attributes are deprecated, e.g. `font`, `center`, etc.
- CSS can be applied to any XML document
  - Not just to HTML / XHTML
- CSS can specify different styles for different media
  - On-screen
  - In print
  - Handheld, projection, etc.
  - … even by voice or Braille-based reader

CSS Introduction

- Håkon Wium Lie – 1994 - Bert Bos



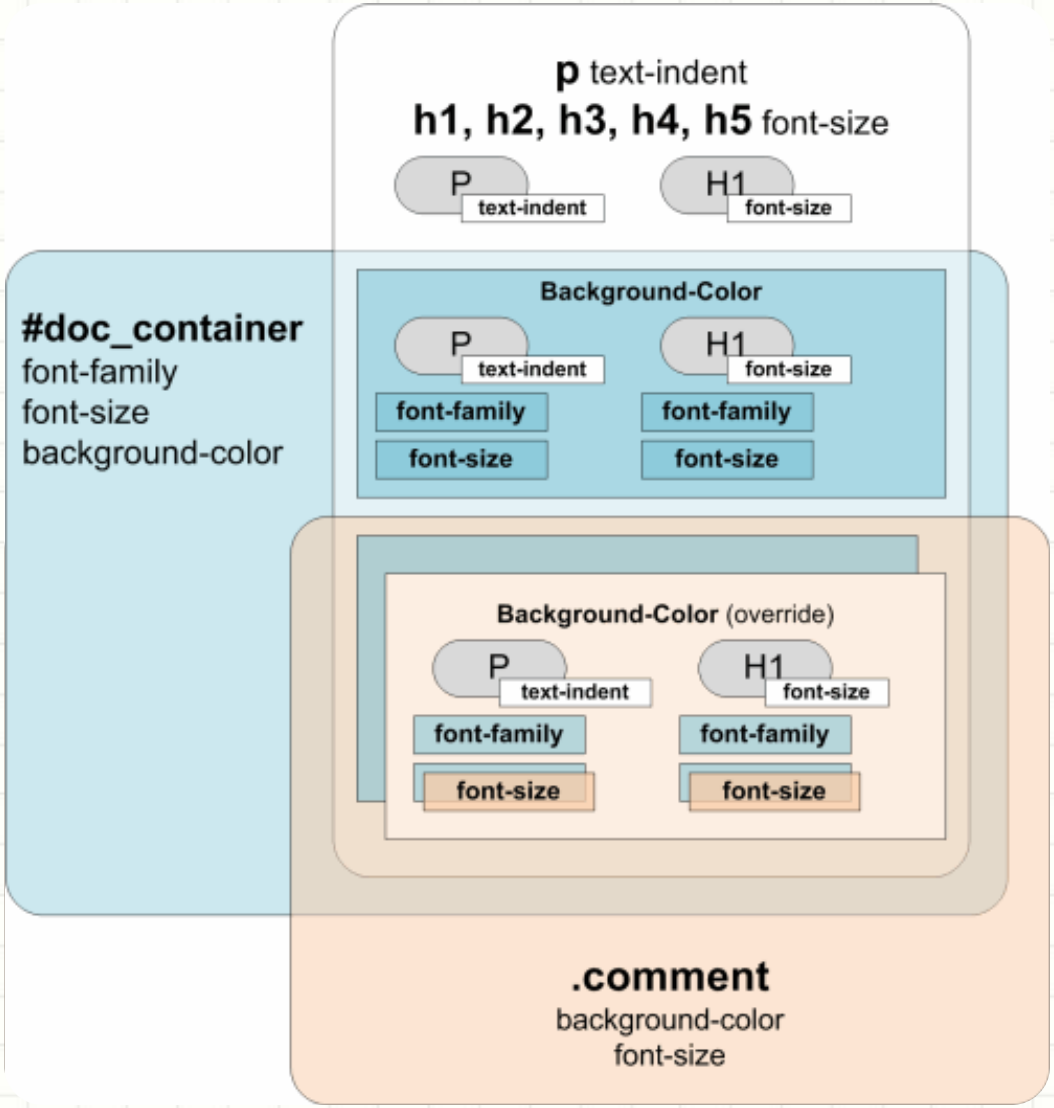
History


CSS 1
CSS 2.1
Selector 3
Selector 4
Borders 3
Images 3
Images 4
Images 5
Multicol
Multicol 2
Grid Layout
Time
Standard CSS as of 199x
(Snapshot 1998)
Standard CSS as of 200x
(Snapshot 200x)
Standard CSS as of 201x
(Snapshot 201x)
Standard CSS as of 201y
(Snapshot 201y)
Standard CSS as of 201z
(Snapshot 201z)

Why “Cascading”?

- Priority scheme determining which style rules apply to element
  - **Cascade priorities** or **specificity (weight)** are calculated and assigned to the rules
  - Child elements in the HTML DOM tree inherit styles from their parent
    - Can override them
    - Control via `!important` rule

Why “Cascading”? (2)



Why “Cascading”? (3)

- Some CSS styles are inherited and some not
  - Text-related and list-related properties are inherited - `color`, `font-size`, `font-family`, `line-height`, `text-align`, `list-style`, etc
  - Box-related and positioning styles are not inherited - `width`, `height`, `border`, `margin`, `padding`, `position`, `float`, etc
  - `<a>` elements do not inherit color and text-decoration

CSS Cascade (Precedence)

- There are browser, user and author stylesheets with "normal" and "important" declarations
  - Browser styles (least priority)
  - Normal user styles
  - Normal author styles (external, in head, inline)
  - Important author styles
  - Important user styles (max priority)

```
a { color: red !important ; }
```

http://www.slideshare.net/maxdesign/css-cascade-1658158

CSS Priorities

- Importance - The ‘!important’ annotation overwrites the previous priority types
- Inline - A style applied to an HTML element via HTML ‘style’ attribute
- Media Type
- User Defined
- Selector specificity
- Rule order
- Parent inheritance
- CSS property definition
- Browser default

CSS Specificity

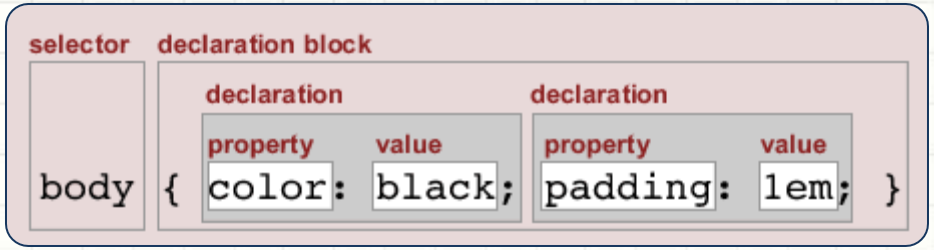
- Universal selectors
- Type selectors
- Class selectors
- Attributes selectors
- Pseudo-classes
- ID selectors
- Inline style

CSS Specificity

- CSS specificity is used to determine the precedence (priority) of the CSS style declarations with the same origin
  - Simple calculation: #id = 100, .class = 10, :pseudo = 10, [attr] = 10, tag = 1, * = 0
  - Same number of points? Order matters!
  - See also:
    - http://www.smashingmagazine.com/2007/07/27/css-specificity-things-you-should-know/
    - http://css.maxdesign.com.au/selectutorial/advanced_conflict.htm

Style Sheets Syntax

- Stylesheets consist of rules, selectors, declarations, properties and values



http://css.maxdesign.com.au/

- Selectors are separated by commas
- Declarations are separated by semicolons
- Properties and values are separated by colons

```
h1,h2,h3 { color: green; font-weight: bold; }
```

Selectors

- Selectors determine which element the rule applies to:
  - All elements of specific type (tag)
  - Those that mach a specific attribute (id, class)
  - Elements may be matched depending on how they are nested in the document tree (HTML)
- Examples:

```
.header a { color: green }
```

```
#menu>li { padding-top: 8px }
```

Primary Selectors

- Three primary kinds of selectors:
  - By tag (type selector):

```
h1 { font-family: verdana,sans-serif; }
```

  - By element id:

```
#element_id { color: #ff0000; }
```

  - By element class name (only for HTML):

```
.myClass {border: 1px solid red}
```

- Selectors can be combined with commas:

```
h1, .link, #top-link {font-weight: bold}
```

This will match <h1> tags, elements with class `link`, and element with id `top-link`

Nested Selectors

- Match relative to element placement:

```
p a {text-decoration: underline}
```

  This will match all <a> tags that are inside of <p>

- * – universal selector (avoid or use with care!):

```
p * {color: black}
```

  This will match all descendants of <p> element

- + selector – used to match “next sibling”:

  This will match all siblings with class name `link` that appear immediately after <`img`> tag

```
img + .link {float:right}
```

Nested Selectors

- > selector – matches direct child nodes:

```
p > .error {font-size: 8px}
```

  This will match all elements with class `error`, direct children of <p> tag

- [ ] – matches tag attributes by regular expression:

```
img[alt~=logo] {border: none}
```

  This will match all <img> tags with `alt` attribute containing the word `logo`

- `.class1.class2` (no space) - matches elements with both (all) classes applied at the same time

Attribute Selectors

- [ ] selects elements based on attributes
  - Element with a given attribute
  - ```
    a[title] {color:black}
    ```

    Selects <a> elements with `title`
  - Elements with a concrete attribute value

    ```
    input[type=text] { font-family:Consolas}
    ```
  - Selects <input> elements with `type=text`
  - Elements whose attribute values contain a word

    ```
    a[title*=logo] {border: none}
    ```
  - Selects <a> elements whose title attribute value contains `logo`
    ```

Common Pseudo Selectors

- Pseudo-classes define state
  - `:hover`, `:visited`, `:active` , `:lang`
- Pseudo-elements define element "parts" or are used to generate content
  - `:first-line` , `:before`, `:after`

```
a:hover { color: red; }
p:first-line { text-transform: uppercase; }
.title:before { content: "»"; }
.title:after { content: "«"; }
```

Structural Pseudo-classes

- `:root`
  - The root of the document
- `E:nth-child(n)`
  - An E element, the n-th child of its parent
- `E:nth-last-child(n)`
  - An E element, the n-th child of its parent, counting from the last on
- `E:nth-of-type(n)`
  - An E element, the n-th sibling of its type

Structural Pseudo-classes (2)

- `E:nth-last-of-type(n)`
  - An E element, the n-th sibling of its type, counting from the last one
- `E:last-child`
  - An E element, last child of its parent
- `E:first-of-type`
  - An E element, first sibling of its type
- `E:last-of-type`
  - An E element, last sibling of its type

Structural Pseudo-classes (3)

- `E:only-child`
  - An E element, only child of its parent
- `E:only-of-type`
  - An E element, only sibling of its type
- `E:empty`
  - An E element that has no children (including text nodes)
- More detailed descriptions:

http://www.w3.org/TR/css3-selectors/#structural-pseudos

The UI Element States Pseudo-Classes

- `E:enabled`
  - A user interface element E which is enabled
- `E:disabled`
  - A user interface element E which is disabled
- `E:checked`
  - A user interface element E which is checked (for instance a radio-button or checkbox)
  - Currently supported only in Opera and IE10 !

Other CSS 3 Selectors

- `E:target`
  - An E element being the target of the referring URI
- `E:not(s)`
  - An E element that does not match simple selector
- `E ~ F`
  - An F element preceded by an E element

Specificity Example

```html
<div id="test">
  <span>Text</span>
</div>
```

```css
div#test span { color: green }
span { color: red }
div span { color: blue }
```

```css
table td    {height: 50px !important;}
.myTable td {height: 50px !important;}
#myTable td {height: 50px !important;}
```

CSS Values

- All values in CSS are strings
  - They can represent values that are not strings
  - I.e. 14px means size 14 pixels
- Colors are set in a red-green-blue format (RGB)
  - Both in hex and decimal

```
li.nav-item {
  color: #44f1e1
}
```

```
li.nav-item {
  color: rgb(68, 241, 255)
}
```

Size Values

- When setting a size (width, height, font-size…) the values are given as numbers
  - Multiple formats / metrics may be used
  - Pixels, ems, e.g. `12px` , `1.4em`
  - Points, inches, centimeters, millimeters
    - E.g. `10pt` , `1in`, `1cm`, `1mm`
  - Percentages, e.g. `50%`
    - Of the size of the container/font size
  - Zero can be used with no unit: `border: 0;`

Color Values

- Colors in CSS can be represented in few ways
  - Using red-green-blue
    - Or red-green-blue-alpha

The opacity values are from `0.0` to `1.0`

```
color: #f1a2ff
color: rgb(241, 162, 255)
color: rgba(241, 162, 255, 0.1)
```

  - Using hue-saturation-light
    - Or hue-saturation-light-alpha

```
color: hsl(291, 85%, 89%);
color: hsl(291, 85%, 89%, 0.1);
```

RGB Colors

- RGB colors are defined with values for red, green and blue int
- Syntax:
  - `#44fa36` – values are in hex
  - `rgb(<red>, <green>, <blue>)` – decim
- The range for `red`, `green` and `blue` is between integers `0` and `255`

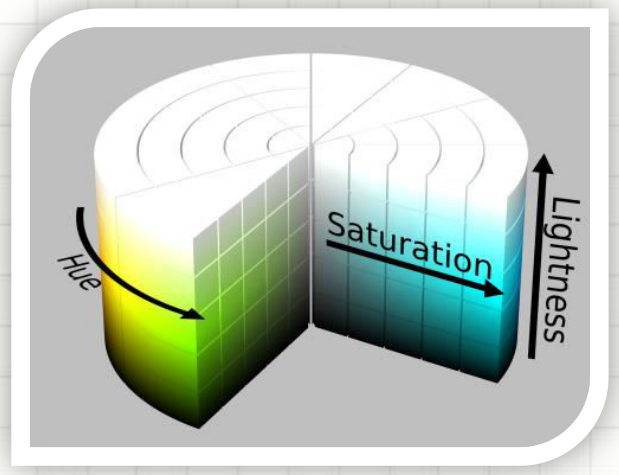
```
color: #07f2b3
  <!- or -->
color: rgb (7, 242, 179)
```

RGBA Colors

- Standard RGB colors with an opacity value for the color (alpha channel)
- Syntax: `rgba(<red>, <green>, <blue>, <alpha>)`
- The range for `red`, `green` and `blue` is between integers `0` ar
- The range for the alpha channel is between `0.0` and `1.0`
- Example: `rgba(255, 0, 0, 0.5)`

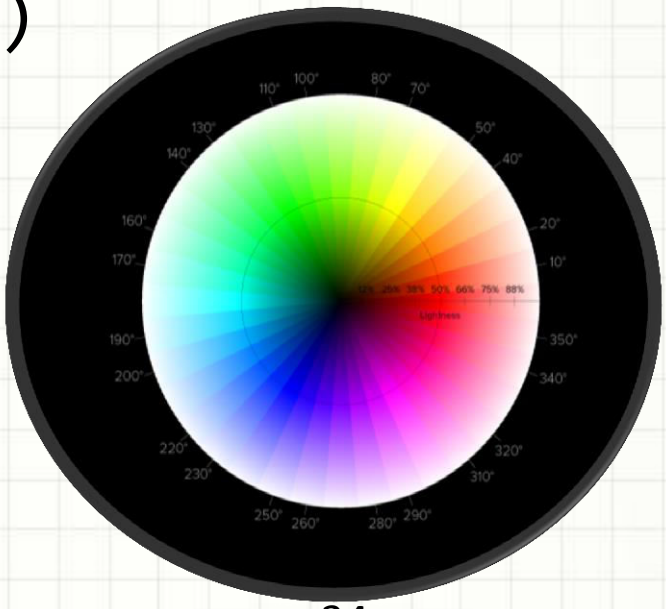
HSL Colors

- Hue is a degree on the color wheel
  - `0` (or `360`) is red, `120` is green, `240` is blue
- Saturation is a percentage value
  - `100%` is the full color
- Lightness is also a percentage
  - `0%` is dark (black)
  - `100%` is light (white)
  - `50%` is the average



HSLA Colors

- HSLA allows a fourth value, which sets the Opacity (via the Alpha channel) of the element
- As RGBA is to RGB, HSLA is to HSL
- Supported in IE9+, Firefox 3+, Chrome, Safari, and in Opera 10+
- Example:
  - `hsla(0, 100%, 50%, 0.5)`
  - Result:



Default Browser Styles

- Browsers have default CSS styles
  - Used when there is no CSS information or any other style information in the document
- Caution: default styles differ in browsers
  - E.g. margins, paddings and font sizes differ most often and usually developers reset them

```
* { margin: 0; padding: 0; }
```

```
body, h1, p, ul, li { margin: 0; padding: 0; }
```

Linking HTML and CSS

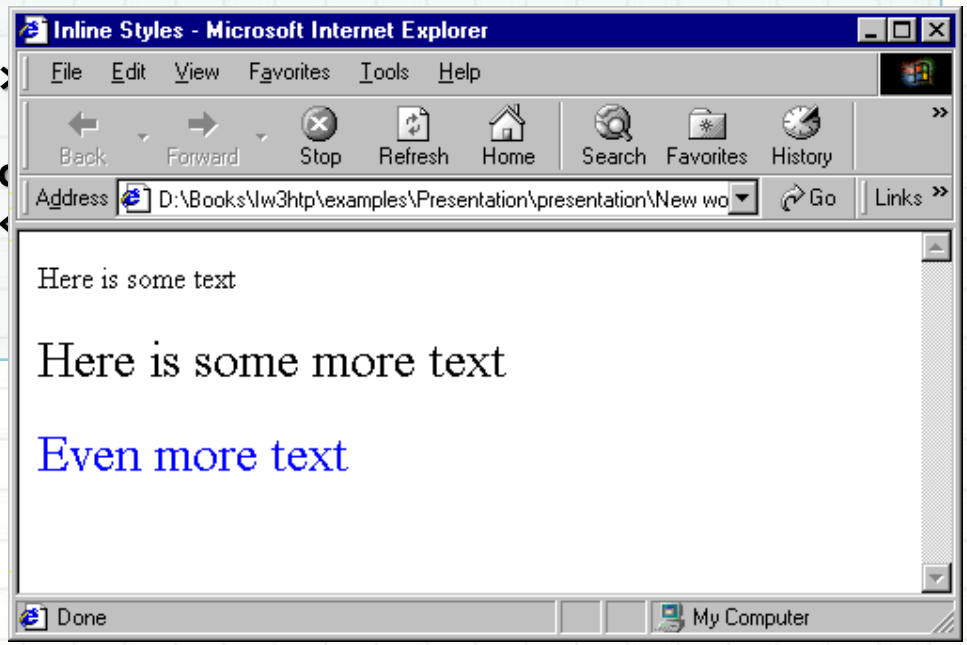
- HTML (content) and CSS (presentation) can be linked in three ways:
  - Inline: the CSS rules in the `style` attribute
    - No selectors are needed
  - Embedded: in the <head> in a `<style>` tag
  - External: CSS rules in separate file (best)
    - Usually a file with `.css` extension
    - Linked via `<link rel="stylesheet" href=…>` tag or `@import` directive in embedded CSS block
- Using external files is highly recommended
  - Simplifies the HTML document
  - Improves page load speed as the CSS file is cached

Inline Styles: Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/ DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Inline Styles</title>
</head>
<body>
  <p>Here is some text</p>
<!--Separate multiple styles with a semicolon-->
  <p style="font-size: 20pt">Here is some
    more text</p>
  <p style="font-size: 20pt;color:
    #0000FF" >Even more text</p>
</body>
</html>
```

Inline Styles: Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/ DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Inline Styles</title>
</head>
<body>
  <p>Here is some text</p>
<!--Separate multiple styles
  <p style="font-size: 20pt";
    more text</p>
  <p style="font-size: 20pt;c
    #0000FF" >Even more text<
</body>
</html>
```



CSS Cascade (Precedence)

- There are browser, user and author stylesheets with "normal" and "important" declarations
  - Browser styles (least priority)
  - Normal user styles
  - Normal author styles (external, in head, inline)
  - Important author styles
  - Important user styles (max priority)

```
a { color: red !important ; }
```

http://www.slideshare.net/maxdesign/css-cascade-1658158

CSS Specificity

- CSS specificity is used to determine the precedence of CSS style declarations with the same origin. Selectors are what matters
  - Simple calculation: #id = 100, .class = 10, :pseudo = 10, [attr] = 10, tag = 1, * = 0
  - Same number of points? Order matters.
  - See also:
  - http://www.smashingmagazine.com/2007/07/27/css-specificity-things-you-should-know/
  - http://css.maxdesign.com.au/selectutorial/advanced_conflict.htm

Embedded Styles

- Embedded in the HTML in the `<style>` tag:

```
<style type="text/css">
```

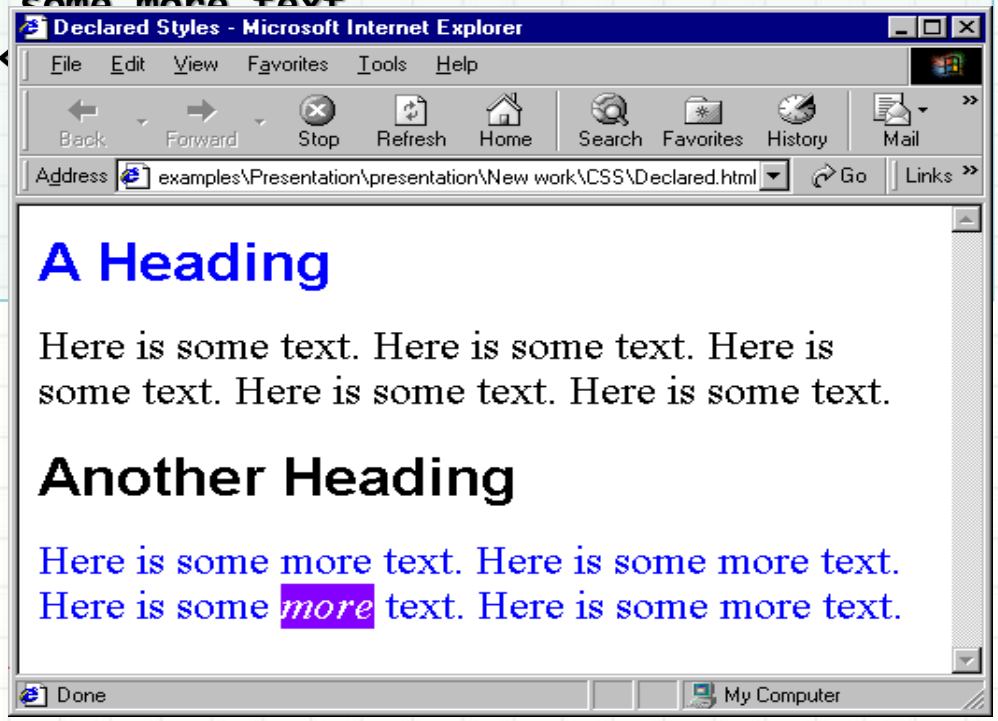
  - The `<style>` tag is placed in the `<head>` section of the document
  - `type` attribute specifies the MIME type
    - MIME describes the format of the content
    - Other MIME types include `text/html`, `image/gif`, `text/javascript` …
- Used for document-specific styles

Embedded Styles: Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Style Sheets</title>
  <style type="text/css">
    em {background-color:#8000FF; color:white}
    h1 {font-family:Arial, sans-serif}
    p  {font-size:18pt}
    .blue {color:blue}
  </style>
<head>
<body>
  <h1 class="blue">A Heading</h1>
  <p>Here is some text. Here is some text. Here
  is some text. Here is some text. Here is some
  text.</p>
  <h1>Another Heading</h1>
  <p class="blue">Here is some more text.
  Here is some more text.</p>
  <p class="blue">Here is some <em>more</em>
  text. Here is some more text.</p>
</body>
</html>
```

Embedded Styles: Example (2)

```
…
<body>
  <h1 class="blue">A Heading</h1>
  <p>Here is some text. Here is some text. Here
  is some text. Here is some text. Here is some
  text.</p>
  <h1>Another Heading</h1>
  <p class="blue">Here is some more text.
  Here is some more text.<
  <p class="blue">Here is
  text. Here is some more
</body>
</html>
```



External CSS Styles

- External linking
  - Separate pages can all use a shared style sheet
  - Only modify a single file to change the styles across your entire Web site (see http://www.csszengarden.com/)
- `link` tag (with a `rel` attribute)
  - Specifies a relationship between current document and another documer

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

  - `link` elements should be in the `<head>`

External CSS Styles (2)

`@import`

- Another way to link external CSS files
- Example:

```
<style type="text/css">
  @import url("styles.css");
  /* same as */
  @import "styles.css";
</style>
```

- Ancient browsers do not recognize @import
- Use @import in an external CSS file to workaround the IE 32 CSS file limit

External Styles: Example

styles.css

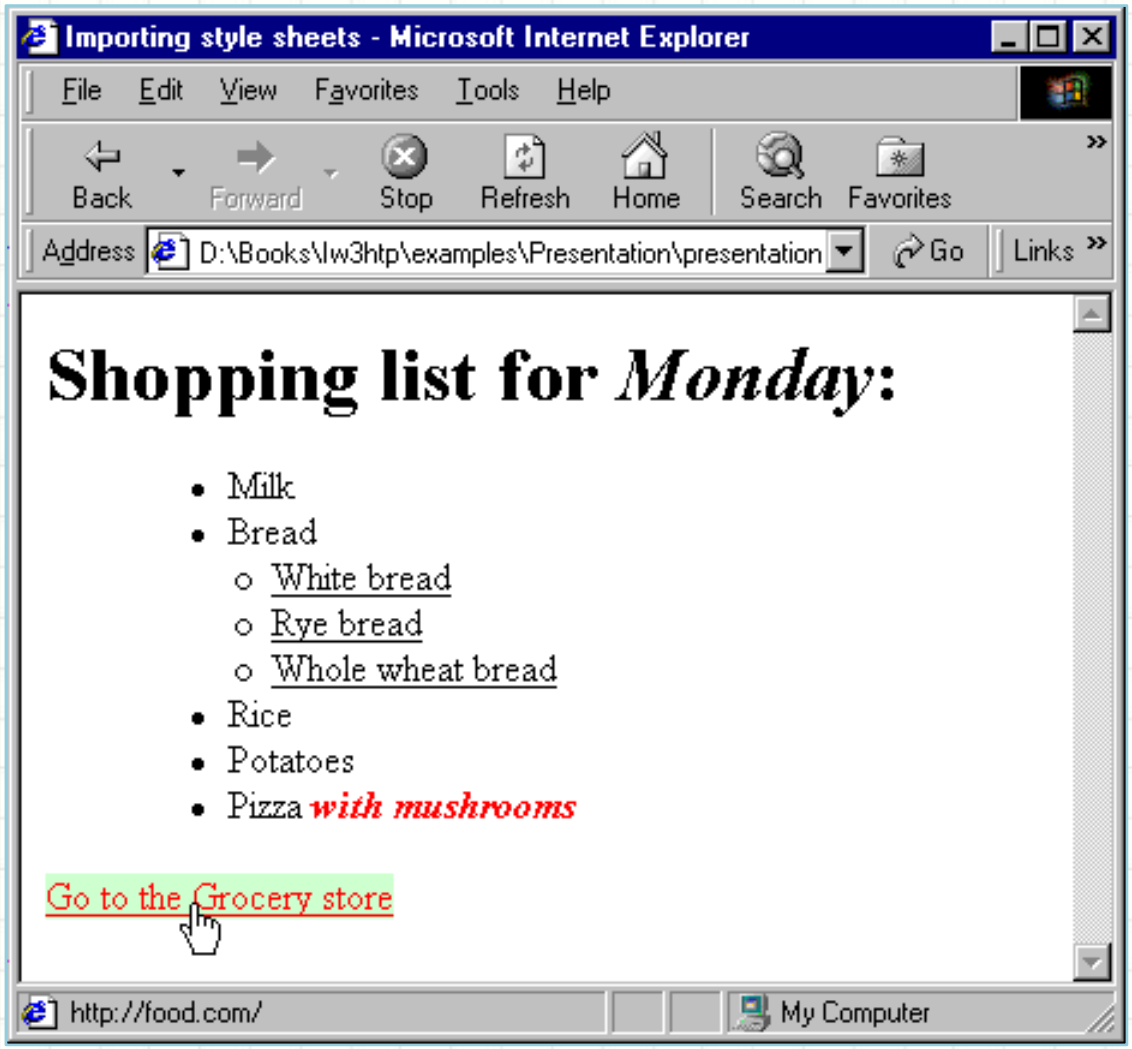
```
/* CSS Document */

a          { text-decoration: none }

a:hover { text-decoration: underline;
          color: red;
          background-color: #CCFFCC }

li em   { color: red;
          font-weight: bold }

ul         { margin-left: 2cm }

ul ul      { text-decoration: underline;
          margin-left: .5cm }
```

External Styles: Example (2)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
  Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Importing style sheets</title>
  <link type="text/css" rel="stylesheet"
    href="styles.css"  />
</head>
<body>
  <h1>Shopping list for <em>Monday</em>:</h1>
  <li>Milk</li>
 <li>Bread
    <ul>
      <li>White bread</li>
      <li>Rye bread</li>
      <li>Whole wheat bread</li>
    </ul>
  </li>
  <li>Rice</li>
  <li>Potatoes</li>
  <li>Pizza <em>with mushrooms</em></li>
</ul>
<a href="http://food.com" title="grocery
  store">Go to the Grocery store</a>
</body>
</html>
```

external-styles.html

External Styles: Example (3)



Text-related CSS Properties

- `color` – specifies the color of the text
- `font-size` – size of font: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`, `smaller`, `larger` or numeric value
- `font-family` – comma separated font names
  - Example: `verdana`, `sans-serif`, etc.
  - The browser loads the first one that is available
  - There should always be at least one generic font
- `font-weight` can be `normal`, `bold`, `bolder`, `lighter` or a number in range [`100` … `900`]

CSS Rules for Fonts (2)

- `font-style` – styles the font
  - Values: `normal`, `italic`, `oblique`
- `text-decoration` – decorates the text
  - Values: `none`, `underline`, `line-trough`, `overline`, `blink`
- `text-align` – defines the alignment of text or other content
  - Values: `left`, `right`, `center`, `justify`

Shorthand Font Property

- font
  - Shorthand rule for setting multiple font properties at the same time

```
font:italic normal bold 12px/16px verdana
```

is equal to writing this:

```
font-style: italic;
font-variant: normal;
font-weight: bold;
font-size: 12px;
line-height: 16px;
font-family: verdana;
```

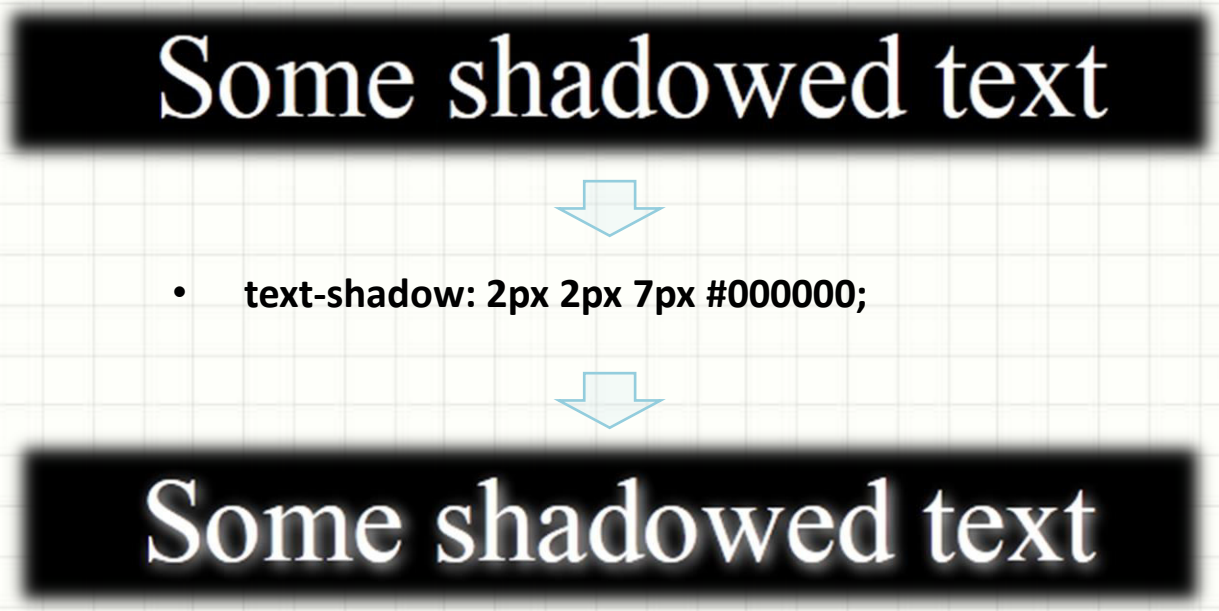
Font Embeds

- Use `@font-face` to declare font
- Point to font file on server
- Call font with font-family
- Currently not supported in IE
- Use font embedding instead of images

```
@font-face {
        font-family: SketchRockwell;
        src: url('SketchRockwell-Bold.ttf');
}
.my_CSS3_class {
        font-family: SketchRockwell;
        font-size: 3.2em;
}
```

Text Shadow

- Applies shadow to text
- Syntax: `text-shadow: <horizontal-distance> <vertical-distance> <blur-radius> <shadow-color>;`
- Do not alter the size of a box



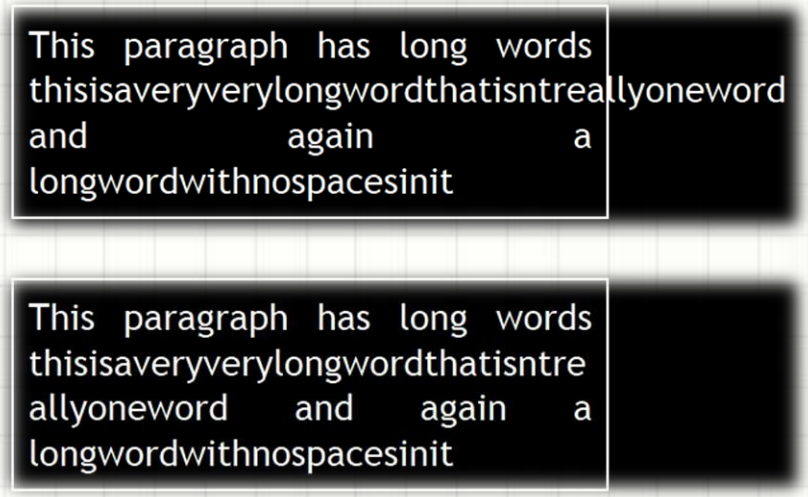
Text Overflow

- Specifies what should happen when text overflows the containing element
- Syntax: `text-overflow: <value>;`
- Possible values:
  - `ellipsis` - Display ellipses to represent clipped text
  - `clip` - Default value, clips text

- Currently not supported in Firefox and IE

Word Wrapping

- Allows long words to be able to be broken and wrap onto the next line
- Syntax: `word-wrap: <value>;`
- Possible values:
  - `normal`
  - `break-word`
- Supported in all major browsers



Backgrounds

- `background-image`
  - URL of image to be used as background, e.g.:

```
background-image:url("back.gif");
```

- `background-color`
  - Using color and image and the same time
- `background-repeat`
  - `repeat-x`, `repeat-y`, `repeat`, `no-repeat`
- `background-attachment`
  - `fixed` / `scroll`
- `background-position`: specifies vertical and horizontal position of the backgroun image
  - Vertical position: `top`, `center`, `bottom`
  - Horizontal position: `left`, `center`, `right`
  - Both can be specified in percentage or other numerical values
  - Examples:

```
background-position: top left;
background-position: -5px 50%;
```

Background Shorthand Property

- background: shorthand rule for setting background properties at the same time:

```
background: #FFF0C0 url("back.gif") no-repeat fixed top;
```

  is equal to writing:

```
background-color: #FFF0C0;
background-image: url("back.gif");
background-repeat: no-repeat;
background-attachment: fixed;
background-position: top;
```

- Some browsers will not apply BOTH color and image for background if using shorthand rule
- Background images allow you to save many image tags from the HTML
  - Leads to less code
  - More content-oriented approach
- All images that are not part of the page content (and are used only for "beautification") should be moved to the CSS

Gradient Backgrounds

- Gradients are smooth transitions between two or more specified colors
- Use of CSS gradients can replace images and reduce download time
  - Lots of gradient generators on the WEB
- Create a more flexible layout, and look better while zooming
- Supported in all major browsers via different keywords
- This is still an experimental feature

Gradient Backgrounds Example

```
/* Firefox 3.6+ */
background: -moz-linear-gradient(100% 100% 90deg,
  #FFFF00, #0000FF);
/* Safari 4-5, Chrome 1-9 */
background: -webkit-gradient(linear, 0% 0%, 0%
  100%, from(#0000FF), to(#FFFF00));
/* Safari 5.1+, Chrome 10+ */
background: -webkit-linear-gradient(#FFFF00,
  #0000FF);
/* Opera 11.10+ */
background: -o-linear-gradient(#2F2727, #0000FF);
```

Multiple Backgrounds

- CSS3 allows multiple background images
- Simple comma-separated list of images
- Supported in Firefox (3.6+), Chrome (1.0/1.3+), Opera (10.5+) and Internet Explorer (9.0+)
- Comma separated list for the other properties

```
background-image: url(sheep.png), url(grass.png);
```

Opacity

- `opacity`: specifies the opacity of the element
  - Floating point number from 0 to 1
  - For old Mozilla browsers use `-moz-opacity`
  - For IE use `filter:alpha(opacity=value)` where value is from 0 to 100; also, "binary and script behaviors" must be enabled and `hasLayout` must be triggered, e.g. with `zoom:1`

Borders

- `border-width`: `thin`, `medium`, `thick` or numerical value (e.g. `10px`)
- `border-color`: color alias or RGB value
- `border-style`: `none`, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset`
- Each property can be defined separately for left, top, bottom and right
  - `border-top-style`, `border-left-color`, …

Border Shorthand Property

- `border`: shorthand rule for setting border properties at once:

```
border: 1px solid red
```

  is equal to writing:

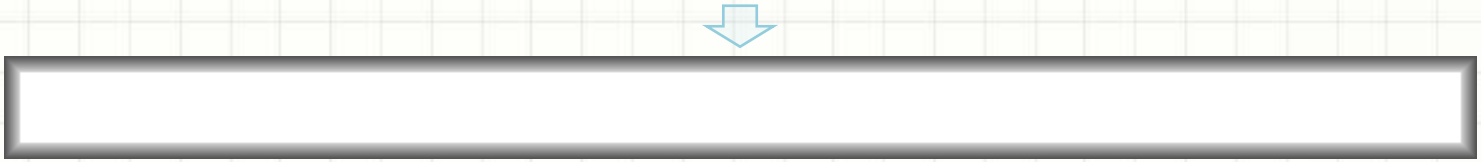
```
border-width:1px;
border-color:red;
border-style:solid;
```

- Specify different borders for the sides via shorthand rules: `border-top`, `border-left`, `border-right`, `border-bottom`
- When to avoid `border:0`

Border color

- Allows you to create cool colored borders
- Only Firefox supports this type of coloring

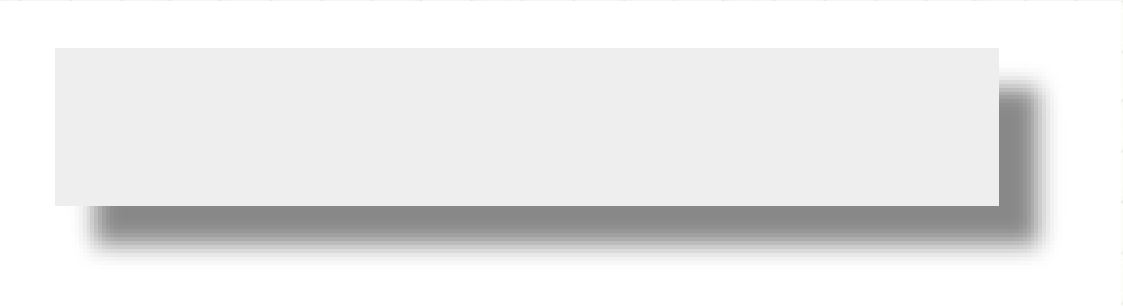
```
border: 8px solid #000;
-moz-border-bottom-colors: #555 #666 #777 #888 #999 #aaa #bbb #ccc;
-moz-border-top-colors: #555 #666 #777 #888 #999 #aaa #bbb #ccc;
-moz-border-left-colors: #555 #666 #777 #888 #999 #aaa #bbb #ccc;
-moz-border-right-colors: #555 #666 #777 #888 #999 #aaa #bbb #ccc;
```



Box shadow

- Allows to easily implement multiple drop shadows (outer or inner) on box elements
- Specifying values for color, size, blur and offset
- Example:

```
-moz-box-shadow: 10px 10px 5px #888;
-webkit-box-shadow: 10px 10px 5px #888;
box-shadow: 10px 10px 5px #888;
```



Rounded Corners

- Rounded corners are a part of CSS 3
  - Supported in all major browsers
  - Firefox, IE 9, Chrome, Opera and Safari
- Done by the `border-radius` property

```
border-radius: [<length>|<%>][<length>|<%>]?
```

- Three ways to define corner radius:

```
border-radius: 15px;
```

```
border-radius: 15px 15px 15px 10px;
```

```
border-radius: 15px 20px;
```

Exercises

1. Create the following page section using HTML and external CSS (no inline styles). Use a table or a definition list (in this case the layout will be different).

Ticket ID: 409788

Name:	Parker Seidel No Active Support
Email:	seidel@aptmail.com
Response Time:	72hours, 60hours left
Last Action:	Never been locked.
	Link to public post
Status:	Not answered

Exercises (2)

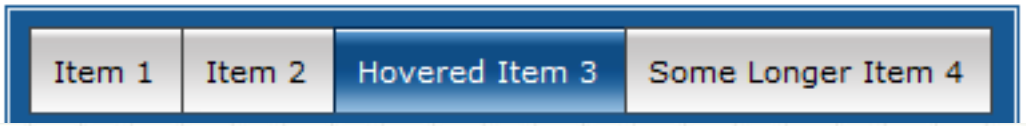
- Create the following Web page using external CSS styles. The country flags should be PNG images with text over them.

Contact name	City	Country	Is Promoted
Maria Anders	Berlin	Germany	☐
Ana Trujillo	México D.F.	Mexico	☐
Antonio Moreno	México D.F.	Mexico	☑
Thomas Hardy	London	UK	☐

Exercises (3)

3. Create the following Web page region using HTML with external CSS file. Note that each of the sections should be a hyperlink.

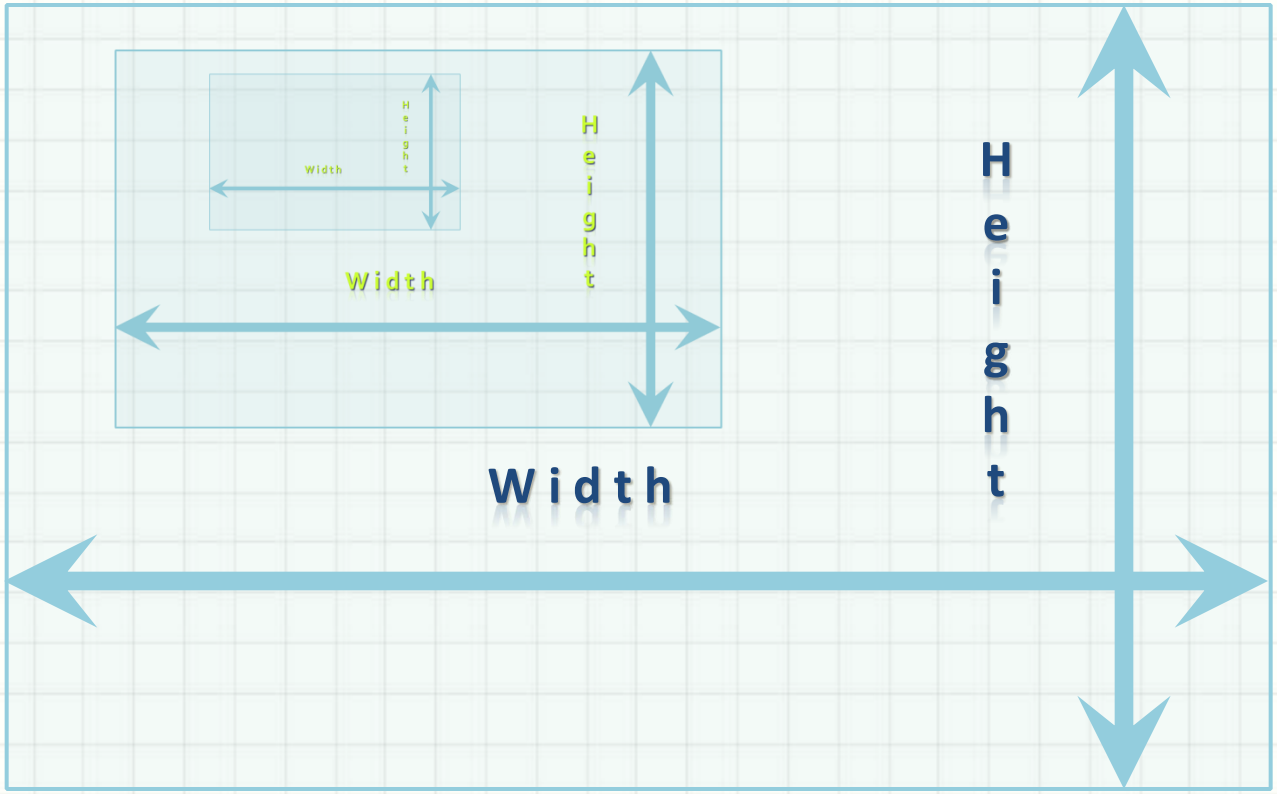
   Hints: use `display:inline-block` style for the list items and paddings where needed.



Width and Height

- `width` – defines numerical value for the width of element, e.g. `200px`
- `height` – defines numerical value for the height of element, e.g. `100px`
  - By default the height of an element is defined by its content
  - Inline elements do not apply height, unless you change their `display` style.

Width



Width

- `width` – defines numerical value for the width of element, e.g. `200px`
- `width` applies only for block elements
  - Their with is 100% by default
  - The width of inline elements is always the width of their content, by concept
- `min-width` - defines the minimal width
  - `min-width` overrides width if (`width<min-width`)
- `max-width` - defines the maximal width
  - `max-width` overrides width if (`width>max-width`)

Width Values

- The values of the width property are numerical:
  - Pixels (px)
  - Centimeters (cm)
  - Or percentages
    - A percent of the available width

Height

- `height` – defines numerical value for the height of element, e.g. `100px`
- `height` applies only on block elements
  - The `height` of inline elements is always the height of their content
- `min-height` - defines the minimal height
  - `min-height` overrides height
- `max-height` - defines the maximal height
  - `max-height` overrides height

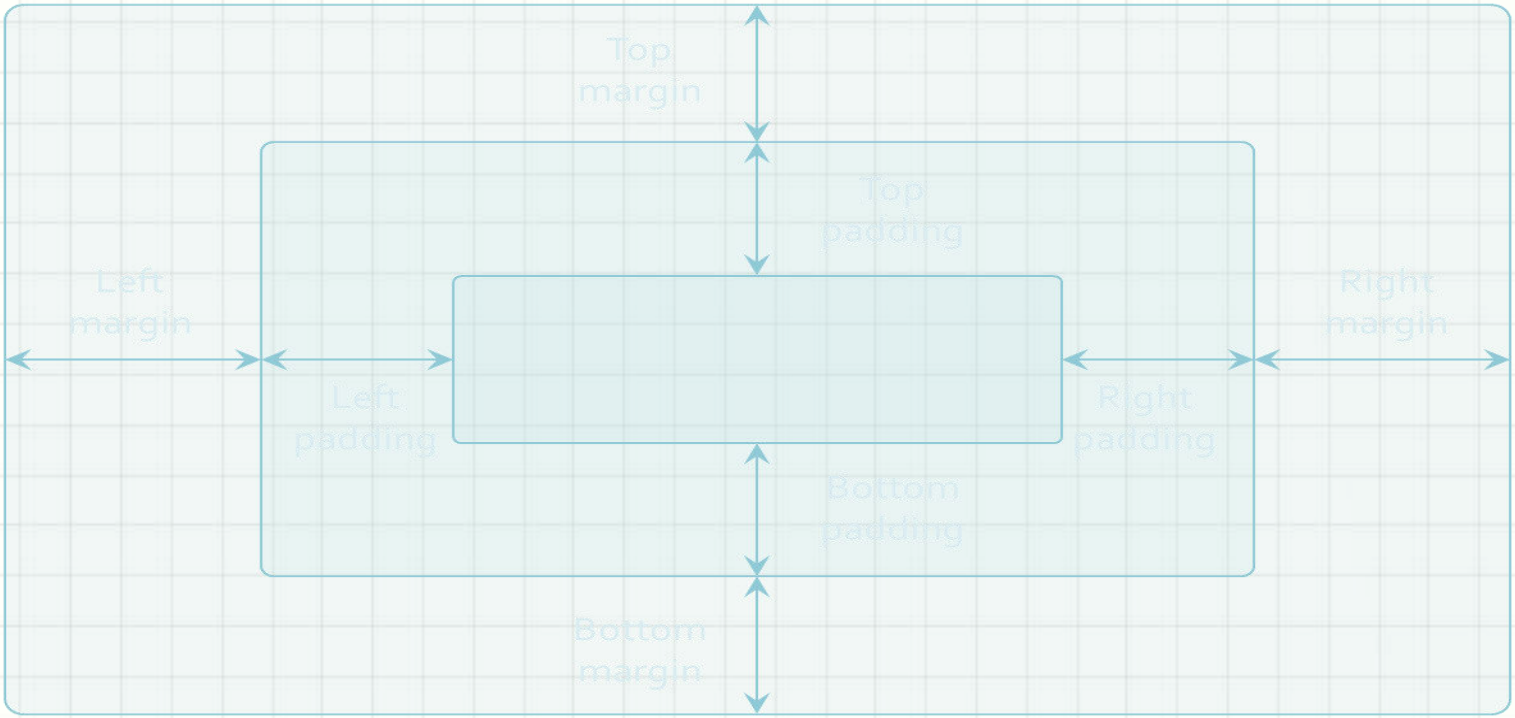
Width / Height

Live Demo

size-rules.html

WHITEBOXQA

Margins and Paddings



Margin and Padding

- `margin` and `padding` define the spacing around the element
  - Numerical value, e.g. `10px` or `-5px`
  - Can be defined for each of the four sides separately - `margin-top`, `padding-left`, …
  - `margin` is the spacing outside of the border
  - `padding` is the spacing between the border and the content
  - What are collapsing margins?

Margin and Padding: Short Rules

- `margin: 5px;`
  - Sets all four sides to have margin of 5 px;
- `margin: 10px 20px;`
  - top and bottom to `10px`, left and right to `20px`;
- `margin: 5px 3px 8px;`
  - top 5px, left/right 3px, bottom 8px
- `margin: 1px 3px 5px 7px;`
  - top, right, bottom, left (clockwise from top)
- Same for `padding`

Margins and Paddings

Live Demo

margins-paddings-rules.html

WHITEBOXQA

Overflow

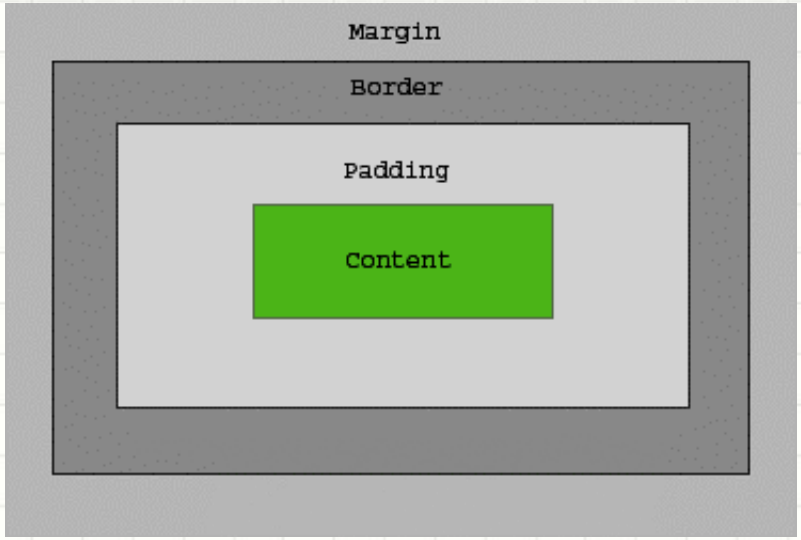
- `overflow`: defines the behavior of element when content needs more space than you have specified by the size properties or for other reasons. Values:
  - `visible` (default) – content spills out of the element
  - `auto` - show scrollbars if needed
  - `scroll` – always show scrollbars
  - `hidden` – any content that cannot fit is clipped

Overflow

overflow-rule.html Live Demo

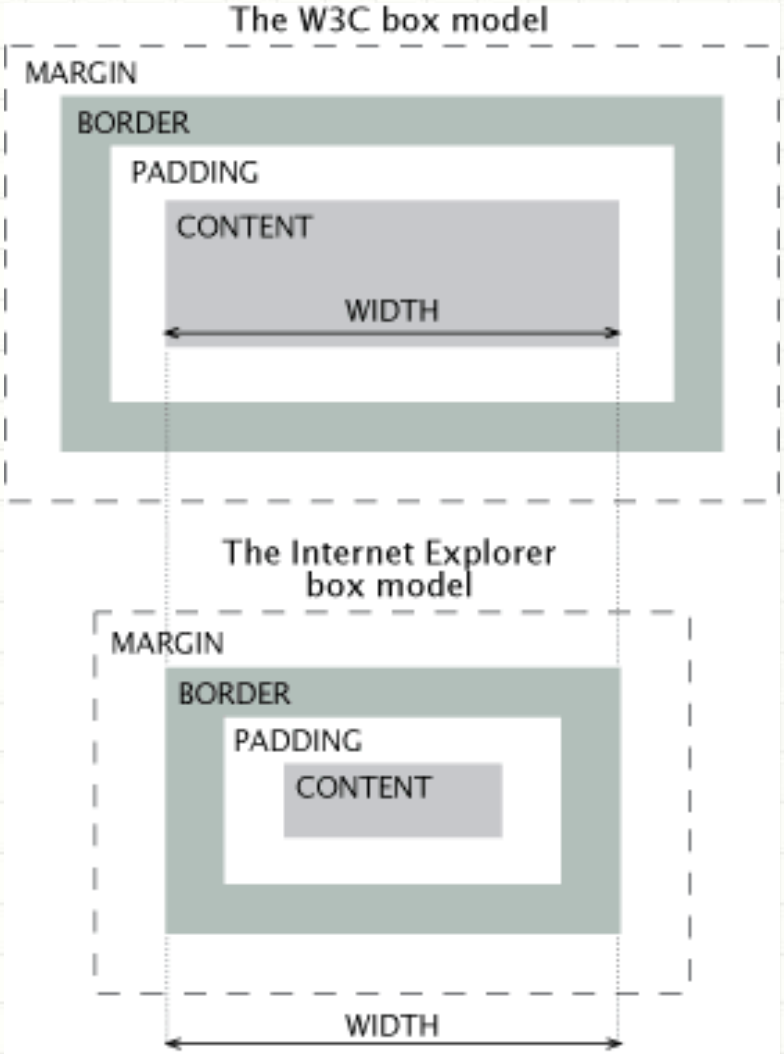
WHITEBOXQA

The Box Model



IE Quirks Mode

- When using quirks mode (pages with no DOCTYPE or with a HTML 4 Transitional DOCTYPE), Internet Explorer violates the box model standard



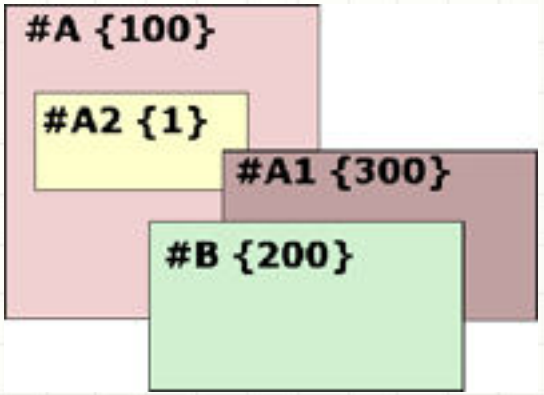
Positioning

- `position`: defines the positioning of the element in the page content flow
- The value is one of:
  - `static` (default)
  - `relative` – relative position according to where the element would appear with static position
  - `absolute` – position according to the innermost positioned parent element
  - `fixed` – same as absolute, but ignores page scrolling
- Margin VS relative positioning
- Fixed and absolutely positioned elements do not influence the page normal flow and usually stay on top of other elements
  - Their position and size is ignored when calculating the size of parent element or position of surrounding elements
  - Overlaid according to their z-index
  - Inline fixed or absolutely positioned elements can apply height like block-level elements

Positioning (2)

- `top`, `left`, `bottom`, `right`: specifies offset of absolute/fixed/relative positioned element as numerical values
- `z-index` : specifies the stack level of positioned elements
  - Understanding stacking context

Each positioned element creates a stacking context.
Elements in different stacking contexts are overlapped according to the stacking order of their containers. For example, there is no way for #A1 and #A2 (children of #A) to be placed over #B without increasing the z-index of #A.



Positioning

Live Demo

positioning-rules.html

WHITEBOXQA

Inline element positioning

- `vertical-align`: sets the vertical-alignment of an inline element, according to the line height
  - Values: `baseline`, `sub`, `super`, `top`, `text-top`, `middle`, `bottom`, `text-bottom` or numeric
- Also used for content of table cells (which apply `middle` alignment by default)

Alignment and Z-Index

Live Demo

alignments-and-z-index-rules.html

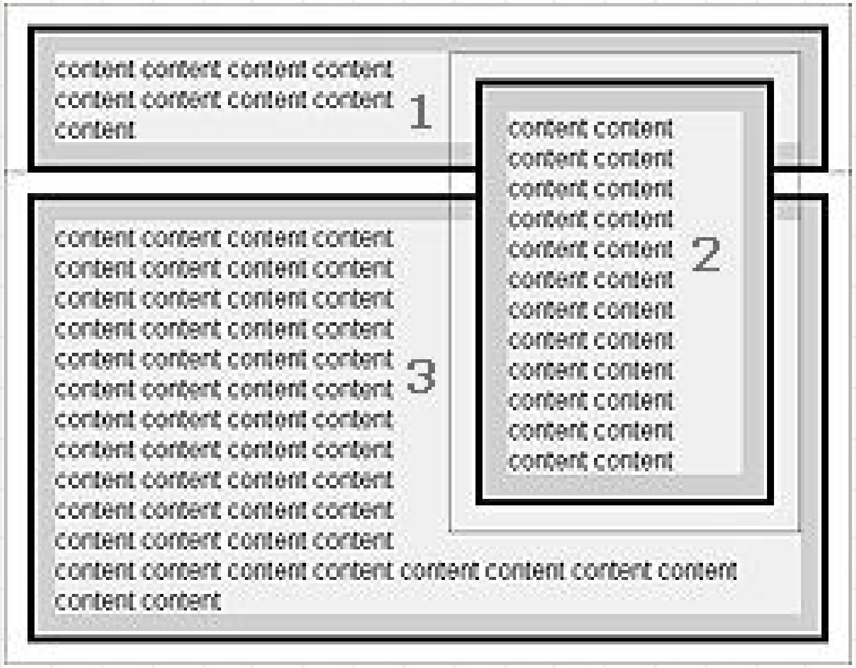
WHITEBOXQA

Float

- `float`: the element "floats" to one side
  - `left`: places the element on the left and following content on the right
  - `right`: places the element on the right and following content on the left
  - floated elements should come before the content that will wrap around them in the code
  - margins of floated elements do not collapse
  - floated inline elements can apply height

Float (2)

- How floated elements are positioned



Clear

- `clear`
  - Sets the sides of the element where other floating elements are NOT allowed
  - Used to "drop" elements below floated ones or expand a container, which contains only floated children
  - Possible values: `left`, `right`, `both`
- Clearing floats
  - additional element (`<div>`) with a clear style
- Clearing floats (continued)
  - `:after { content: ""; display: block; clear: both; height: 0; }`
  - Triggering hasLayout in IE expands a container of floated elements
    - display: inline-block;
    - zoom: 1;

Floating Elements

float-rules.html Live Demo

WHITEBOXQA

Opacity

- `opacity`: specifies the opacity of the element
  - Floating point number from 0 to 1
  - For old Mozilla browsers use `-moz-opacity`
  - For IE use `filter:alpha(opacity=value)` where value is from 0 to 100; also, "binary and script behaviors" must be enabled and `hasLayout` must be triggered, e.g. with `zoom:1`

Opacity

opacity-rule.html Live Demo

WHITEBOXQA

Visibility

- `visibility`
  - Determines whether the element is visible
  - `hidden`: element is not rendered, but still occupies place on the page (similar to `opacity:0`)
  - `visible`: element is rendered normally

Visibility

visibility-rule.html Live Demo

WHITEBOXQA

Display

- `display`: controls the display of the element and the way it is rendered and if breaks should be placed before and after the element
  - `inline`: no breaks are placed before and after (`<span>` is an inline element)
  - `block`: breaks are placed before AND after the element (`<div>` is a block element)
- `display`: controls the display of the element and the way it is rendered and if breaks should be placed before and after the element
  - `none`: element is hidden and its dimensions are not used to calculate the surrounding elements rendering (differs from `visibility: hidden`!)
  - `inline-block`: : no breaks are placed before and after (like inline)
    - Height and width can be applied (like block)
  - There are some more possible values, but not all browsers support them
    - Specific displays like `table-cell` and `table-row`

Display

display-rule.html

Live Demo

WHITEBOXQA

Other CSS Properties

- `cursor`: specifies the look of the mouse cursor when placed over the element
  - Values: `crosshair`, `help`, `pointer`, `progress`, `move`, `hair`, `col-resize`, `row-resize`, `text`, `wait`, `copy`, `drop`, and others
- `white-space` – controls the line breaking of text. Value is one of:
  - `nowrap` – keeps the text on one line
  - `normal` (default) – browser decides whether to brake the lines if needed

Benefits of using CSS

- More powerful formatting than using presentation tags
- Your pages load faster, because browsers cache the `.css` files
- Increased accessibility, because rules can be defined according given media
- Pages are easier to maintain and update

Maintenance Example

CSS file

CSS Development Tools

- WebStorm
- Dreamweaver
- Firebug
- Any IDE

CSS Reference

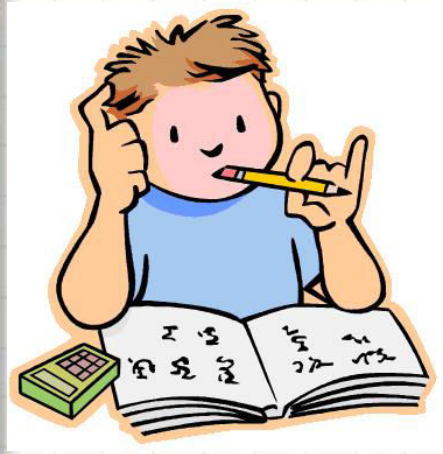
- A list of all CSS 2.1 properties is available at http://www.w3.org/TR/CSS2/propidx.html

Questions

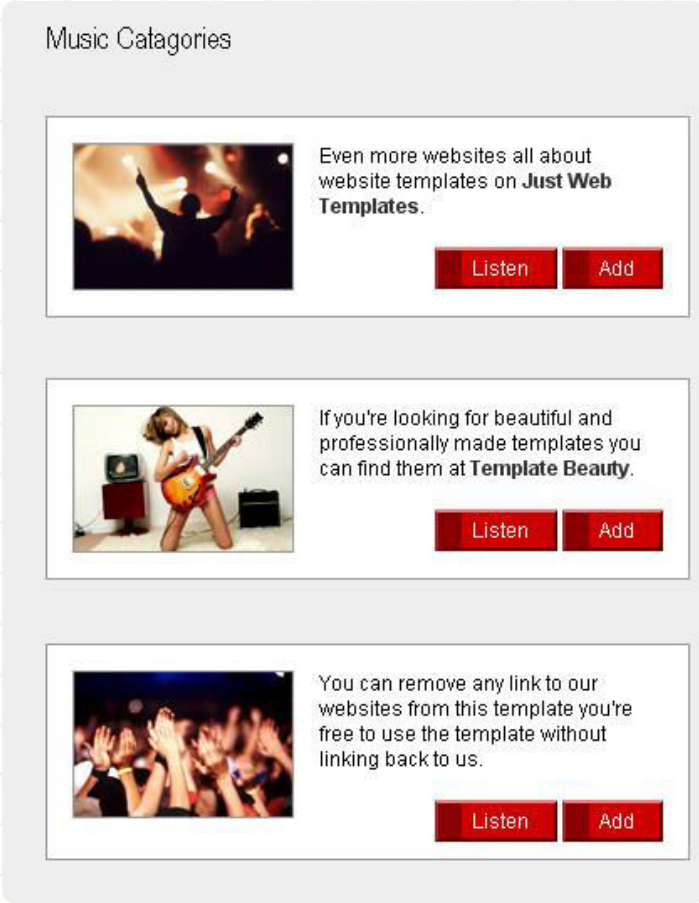
Extensions

- Firebug plug-in for Firefox
  - A must have for Web developers
  - The ultimate tool for monitoring, editing and debugging HTTP, HTML, CSS, JavaScript, etc.
  - Free, open-source – www.getfirebug.com
- EditCSS
  - Edit CSS
  - https://addons.mozilla.org/en-US/firefox/addon/editcss/
- Mozilla Developer Network
  - https://developer.mozilla.org/

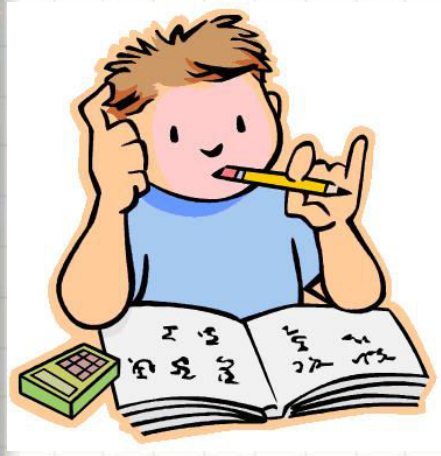
Exercises



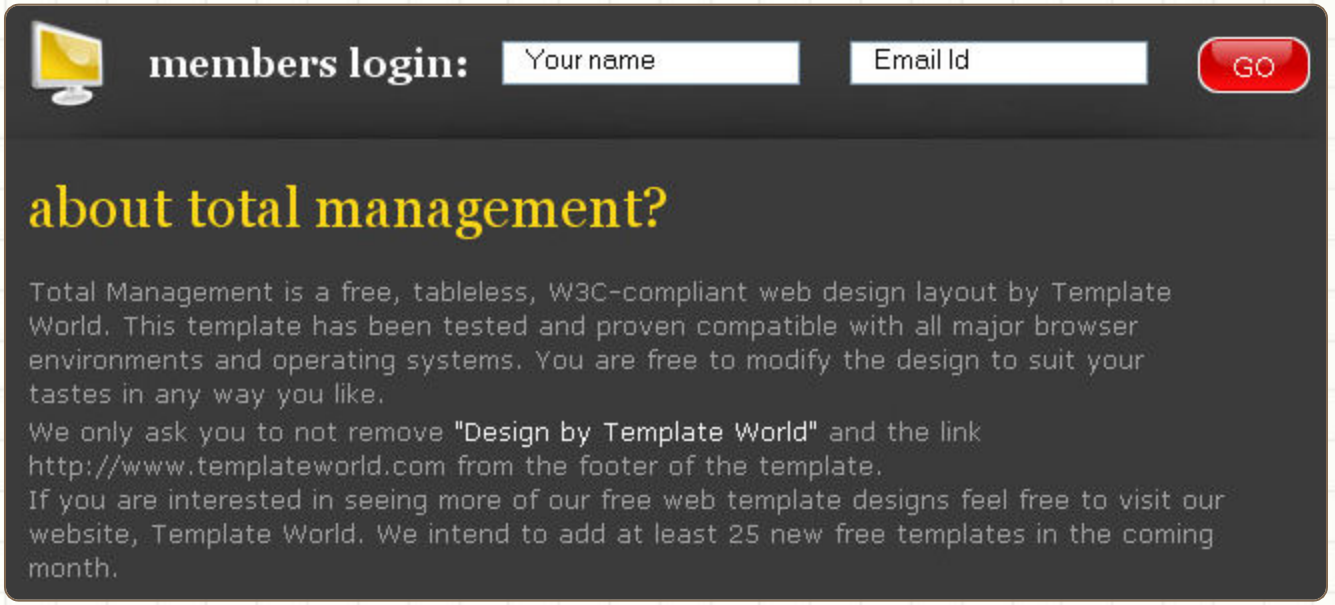
- Create the following Web page using external CSS styles. Buttons should consist of PNG images with text over them.



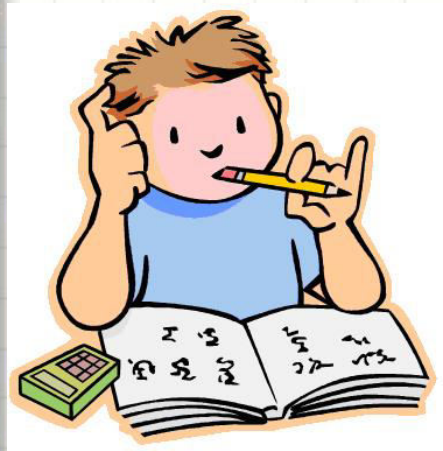
Exercises



- Create the following Web page using HTML with external CSS file. Note that the images should be PNG with transparent background.



Exercises



5. Given the picture below (`CSS-Web-Site.png`) create the Web site. Use CSS and XHTML.