CSS
2
CSSEDIT 2
macrabbit
ie sucks

Acknowledgement

- Some of the slides are based on '*HTTP 5 and CSS 3 Course*' prepared by **Telerik School Academy**

http://downloads.academy.telerik.com/svn/school-academy/Meeting-20-Web-Design-HTML5-CSS3/

Table of Contents

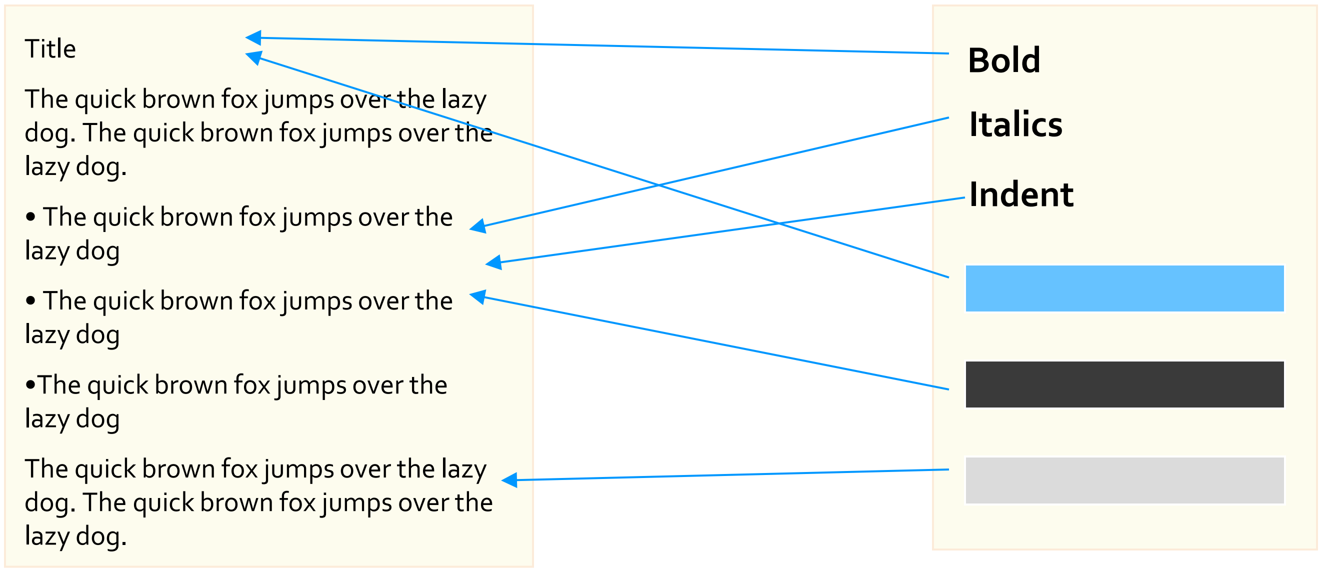
- What is CSS?
- Styling with Cascading Stylesheets (CSS)
- Selectors and style definitions
- Linking HTML and CSS
- Text styles
- List styles
- Margins, Borders and Padding
- Positioning Elements

Why CSS?

- Separate content from presentation!

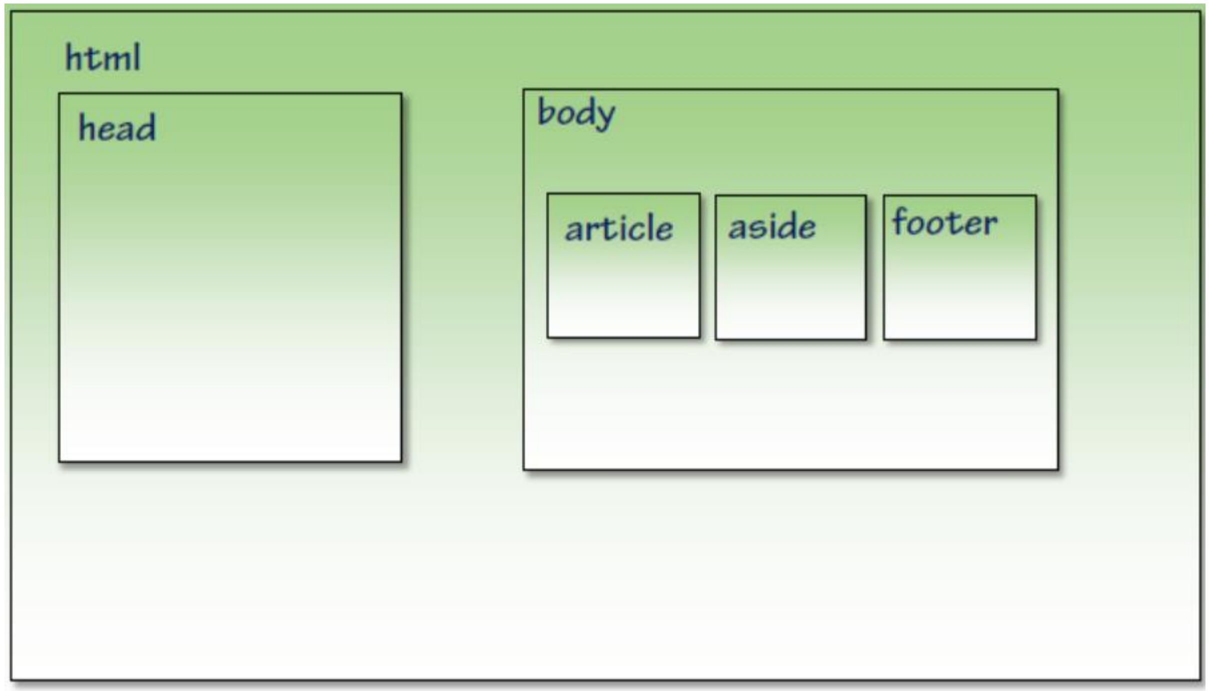
Content (HTML document)

Presentation (CSS Document)


Title
The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.
• The quick brown fox jumps over the lazy dog
• The quick brown fox jumps over the lazy dog
•The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.
Bold
Italics
Indent

CSS Introduction

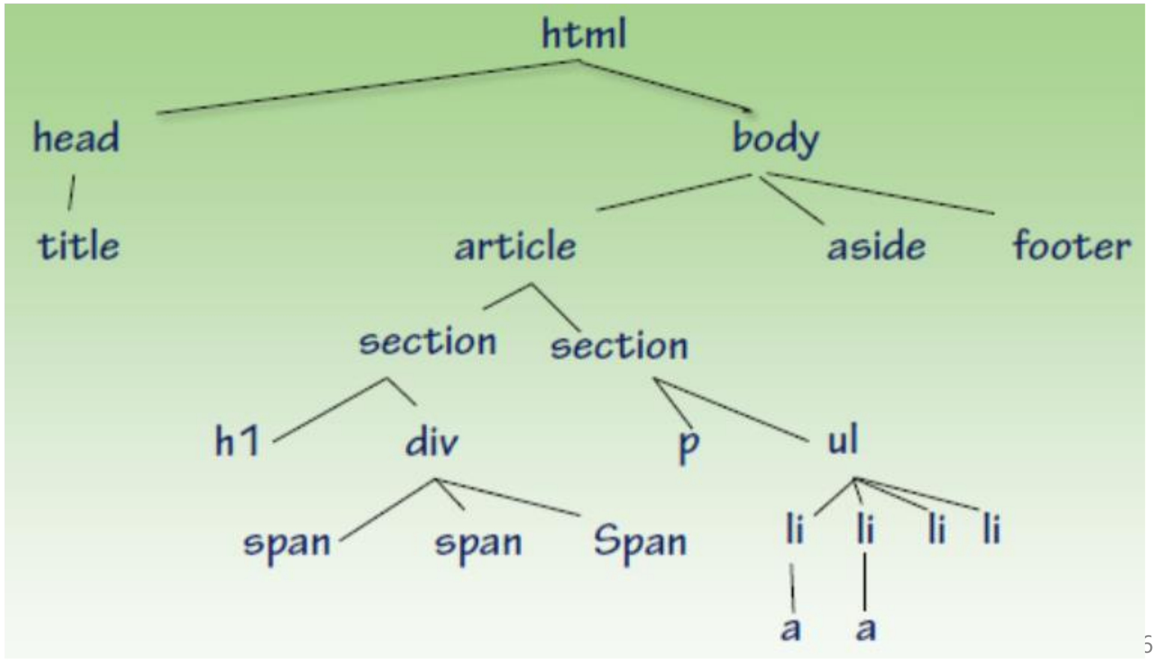
- Cascading Style Sheets (CSS)
  - Used to describe the presentation of documents
  - Define sizes, spacing, fonts, colors, layout, etc.
- Designed to separate presentation from content
  - Ensure consistent look and feel
  - Improve flexibility and maintainability
- Due to CSS, all HTML presentation tags and attributes are deprecated, e.g. `font`, `center`, etc.



An HTML document in represented in the browser as a Document Object Model (DOM) Tree

Relationships between elements

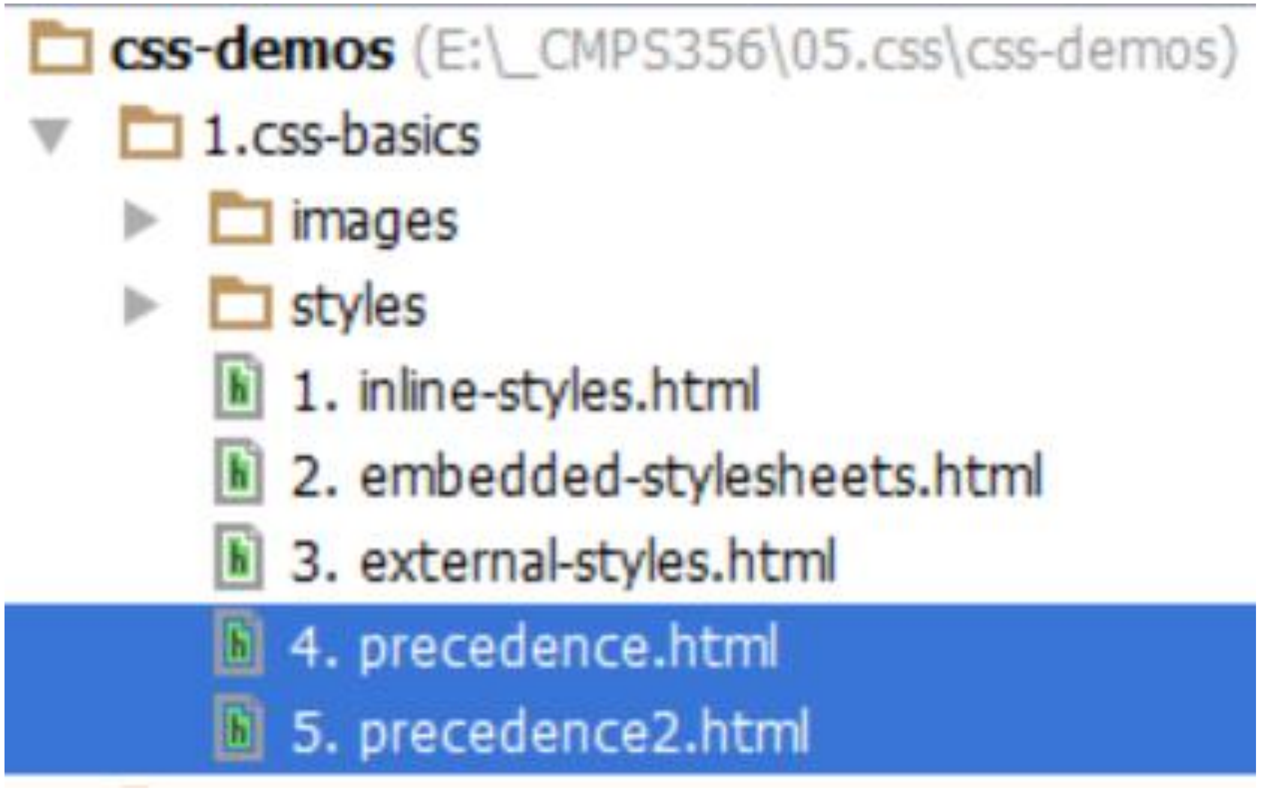
- Ancestor
- Descendent
- Parent
- Child
- Sibling



Why “Cascading”?

- Descendant elements in the HTML DOM tree **inherit** styles from their Ancestors
  - **Nearest ancestor wins**
  - Child can override the inherited style (More specific styles win over inherited styles)
- Cascading saves time and effort
- Text-related and list-related properties are inherited - `color`, `font-size`, `font-family`, `line-height`, `text-align`, `list-style`, etc.
- Some CSS styles are not inherited
  - Box-related and positioning styles are not inherited - `width`, `height`, `border`, `margin`, `padding`, `position`, `float`, background colors, etc.
  - `<a>` elements do not inherit color and text-decoration

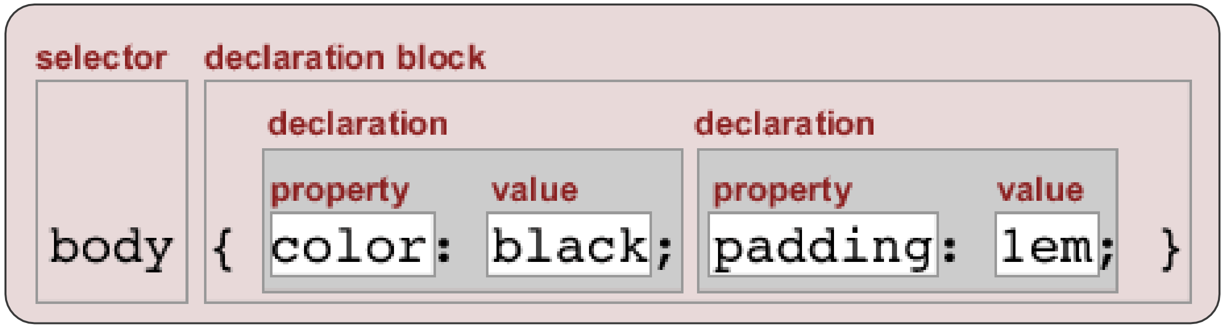
CSS Rules Precedence - Examples



```
css-demos (E:\_CMPS356\05.css\css-demos)
▼ 1.css-basics
   ▶ images
   ▶ styles
     1. inline-styles.html
     2. embedded-stylesheets.html
     3. external-styles.html
     4. precedence.html
     5. precedence2.html
```

Style Sheets Syntax

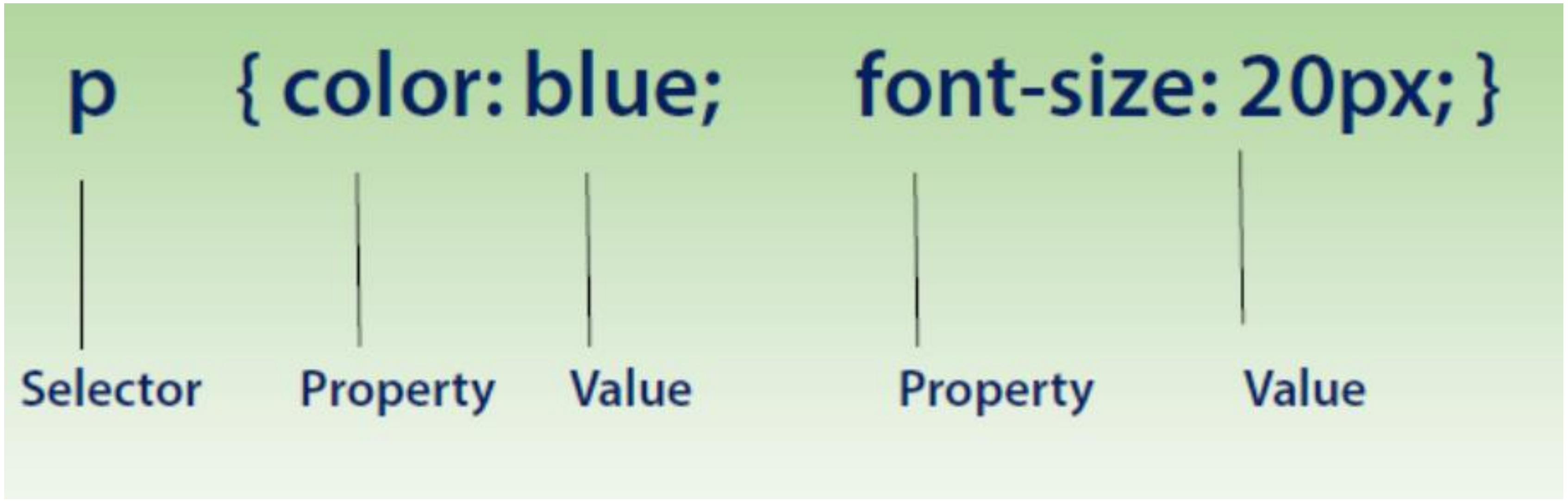
- Stylesheets consist of rules. Each rule has selectors and declarations. A declaration specify a property and its value.



- 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; }
```

Example



Selectors are used to select elements on an HTML page so that they can be styled

Basic Selectors

- **Tag Selectors**
  - Apply page-wide

  e.g., `p { font-family: verdana; }` applies the style to all <p> tags

- **Class Selectors**
  - Defines a **named** style (prefix the name with dot (.))
  - Can apply to any page element using the class attribute

  e.g., `.redBorder {border: 1px solid red}` defines a style named redBorder

  `<p class='redBorder'>Using the class attribute to apply the redBoder style to this paragrpah</p>`

- **ID Selectors**
  - Apply to one specific tag
  - Use hash (#) followed by the tag id to select the element to be styled
  - Good for linking to specific part of a page

  e.g., `#errorMsg { color: red; }` apply the style to the element with id **errorMsg**

Selector Examples

- Three primary kinds of selectors:
  - Tag selectors(aka Type selectors):

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

  - Class Selectors:

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

  - ID Selectors:

```
#errorMsg { color: red; }
```

- Selectors can be combined with commas:

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

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

- Comment in CSS `/* comment */`

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 tag

```
<link rel="stylesheet" href="…">
```

    - Or via `@import` directive in embedded CSS block

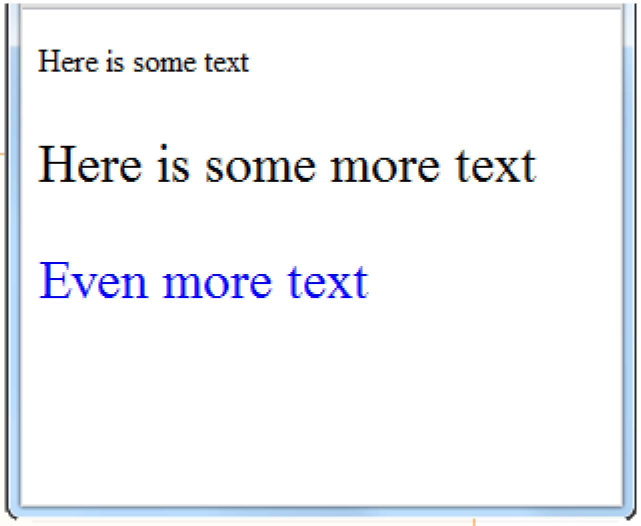
Inline Styles

- CSS rules in the element’s `style` attribute
- No need for selectors
  – Acts on the element on which it is set
- Not recommended
  – It mixes content with presentation
  – The CSS idea is to avoid that mixing

Example - Inline Styles

Example: inline-styles.html

```
<!DOCTYPE html>
<html lang="en">
<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>
```



Here is some text

Here is some more text

Even more text

Example - Embedded Styles

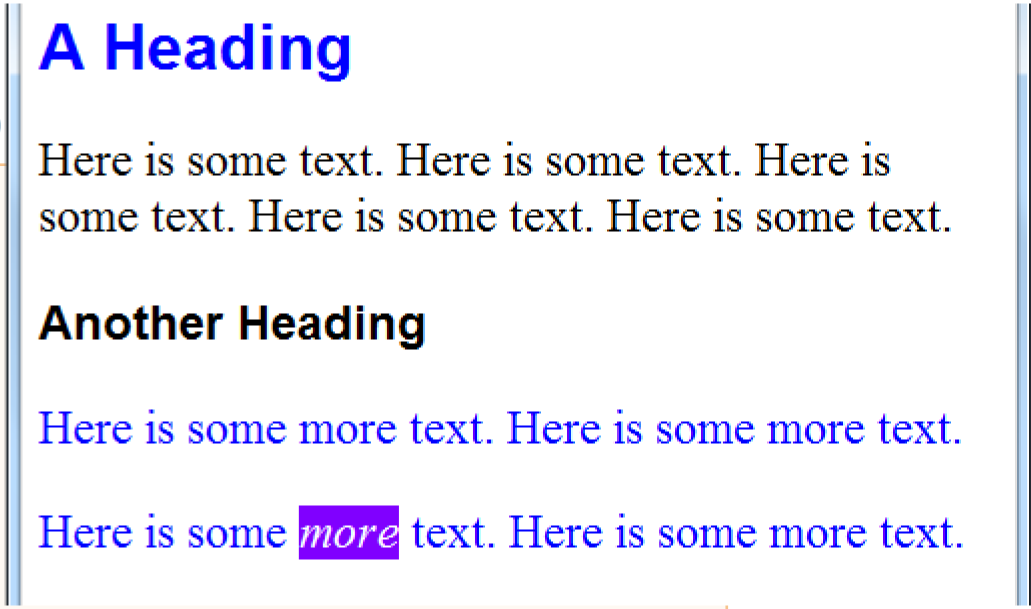
- Used for document-specific styles

Example: embedded-stylesheets.html

```
<!DOCTYPE html>
<html>
<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>
```

Example
Embedded Styles (cont.)

```
…
<body>
  <header>
      <h1 class="blue">
        A Heading
      </h1>
  </header>
  <article>
      <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>
  </article>
</body>
</html>
```


A Heading

Here is some text. Here is some text. Here is some text. Here is some text. Here is some text.

Another Heading

Here is some more text. Here is some more text.

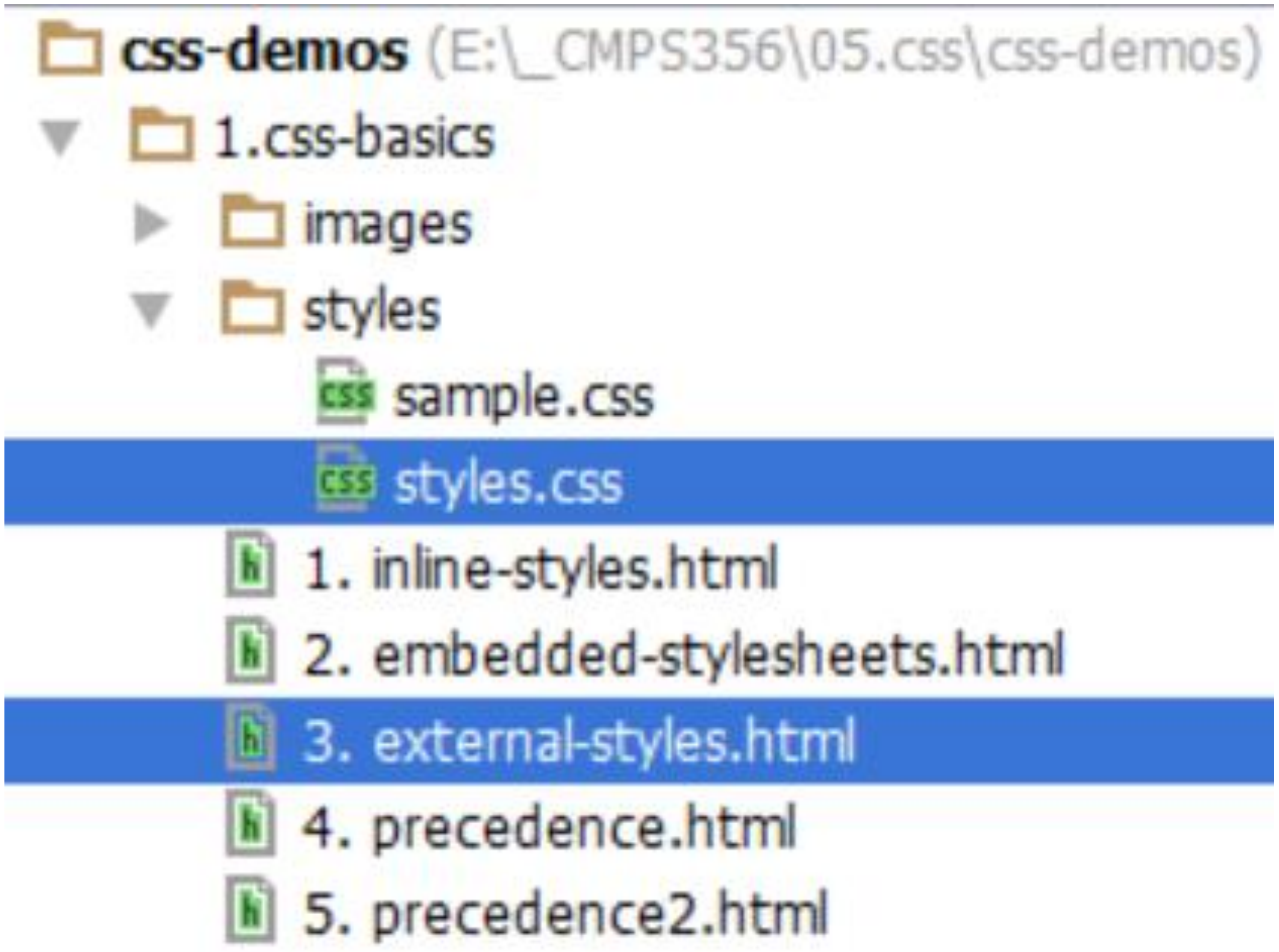
Here is some *more* text. Here is some more text.

External CSS Styles

- Using external files is highly recommended
  - **Separation of concerns**: separates content from presentation
    - Simplifies the HTML document
    - Increases reusability
    - Eases maintainability
  - Insures consistent look and feel across the entire website
    - Only modify a single file to change the styles across the entire Web site
  - Faster page loading as the CSS file is cached
- `<link>` tag in the document `<head>` to link html document with ccs document

```
<link rel="stylesheet" href="styles.css">
```

Example - External Styles



```
css-demos (E:\_CMPS356\05.css\css-demos)
▼ 1.css-basics
    ▶ images
    ▼ styles
        sample.css
        styles.css
      1. inline-styles.html
      2. embedded-stylesheets.html
      3. external-styles.html
      4. precedence.html
      5. precedence2.html
```

Selectors

Combined Selectors

- Match relative to element placement:

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

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

- `+` selector is used to match “next sibling”:

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

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

- `>` selector – matches direct child nodes:

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

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

Attribute Selectors

- `E[foo^="bar"]`
  - An `E` element whose "foo" attribute value begins with the string "bar"
  - Example: `a[src^="https://"]`
- `E[foo$="bar"]`
  - An `E` element whose "foo" attribute value ends with the string "bar"
- `E[foo*="bar"]`
  - An `E` element whose "foo" attribute value contains the substring "bar"

Pseudo-classes

- Pseudo-classes define state
  - `:hover`, `:visited`, `:active`
- 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)

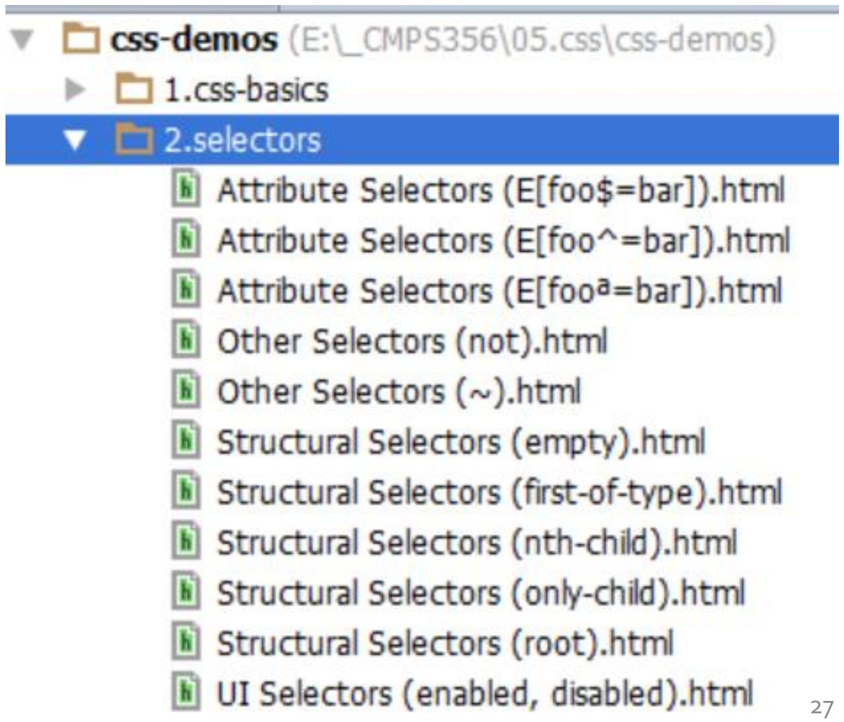
- `E:only-child`
  - An `E` element, only child of its parent
- `E:only-of-type`
  - An `E` element, only sibling of its type
- `E:empty`
  - An `E` element that has no children (including text nodes)
- More detailed descriptions:

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

Summary

- A style consists of a selector, followed by property/value pairs
- Selectors:
  - Tag Selectors
  - Class Selectors
  - ID Selectors
  - Combined Selectors
  - Attribute selectors
  - Pseudo-elements
  - Structural pseudo-classes
  - UI state pseudo-classes

Examples



```
▼ css-demos (E:\_CMPS356\05.css\css-demos)
    ▶ 1.css-basics
    ▼ 2.selectors
        Attribute Selectors (E[foo$=bar]).html
        Attribute Selectors (E[foo^=bar]).html
        Attribute Selectors (E[fooª=bar]).html
        Other Selectors (not).html
        Other Selectors (~).html
        Structural Selectors (empty).html
        Structural Selectors (first-of-type).html
        Structural Selectors (nth-child).html
        Structural Selectors (only-child).html
        Structural Selectors (root).html
        UI Selectors (enabled, disabled).html
```

Values in the CSS Rules

- Colors are set in RGB format (decimal or hex):
  - Example: `#a0a6aa = rgb(160, 166, 170)`
  - Predefined color aliases exist: `black`, `blue`, etc.
- Numeric values are specified in:
  - Pixels, ems, e.g. `12px` , `1.4em`
  - Points, inches, centimeters, millimeters
    - E.g. `10pt` , `1in`, `1cm`, `1mm`
  - Percentages, e.g. `50%` `(e.g., div can occupy 50% of available space)`
  - Zero can be used with no unit: `border: 0;`

Default Browser Styles

- Browsers have predefined 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
  - Usually developers reset them

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

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

Text Styles

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:

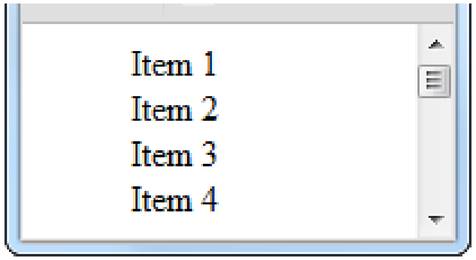
```
font-style: italic;
font-variant: normal;
font-weight: bold;
font-size: 12px;
line-height: 16px;
font-family: verdana;
```

List Styles

Styles for Lists

- List properties are used to define the look and feel of the list items
  - Mainly to change the list item marker
- Normal styles:
  - `List-style-type`
    - Values for <ol>: `circle`, `square`,...
    - Values for <ul>: `upper-roman`, `lower-alpha`
    - Values for both: `none`

```
ul
{
    list-style-type:none;
}
```


Item 1
Item 2
Item 3
Item 4

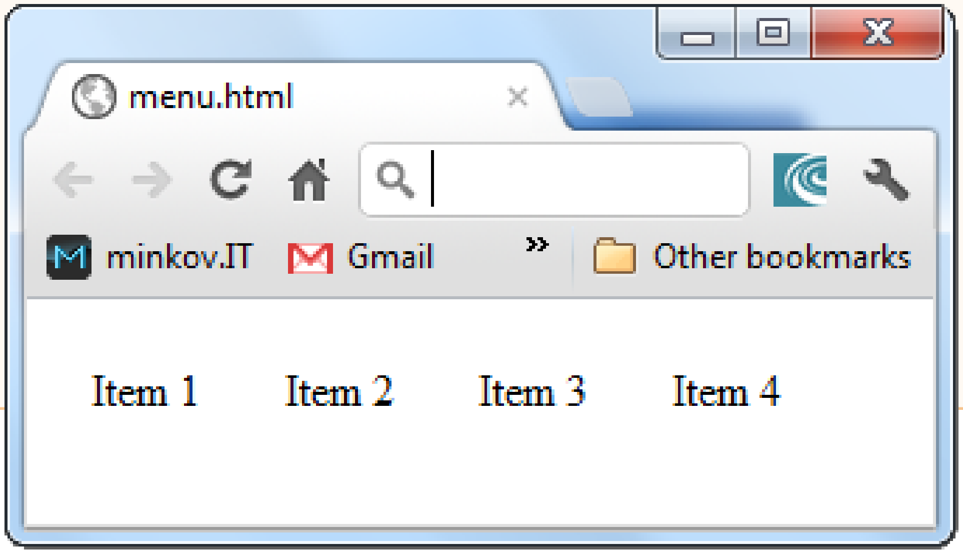
Create Navigation Bar

- A Navigation bar is a set of links
  - A list of the different areas on your site
- Place the link in Unordered List <ul>
- Remove the bullets (list-style-type: none;)
- Eliminate padding and margins
- Set the display to inline-block to eliminate new lines
- Style the links
  - Remove the underline
  - Set the color
  - Surround with a border

Creating a Menu-like List

```
ul.menu
{
   list-style-type: none;
   padding: 0px;
   margin: 0px;
}

li.menu-item
{
   float: left;
   margin: 10px;
}
```



Item 1 Item 2 Item 3 Item 4

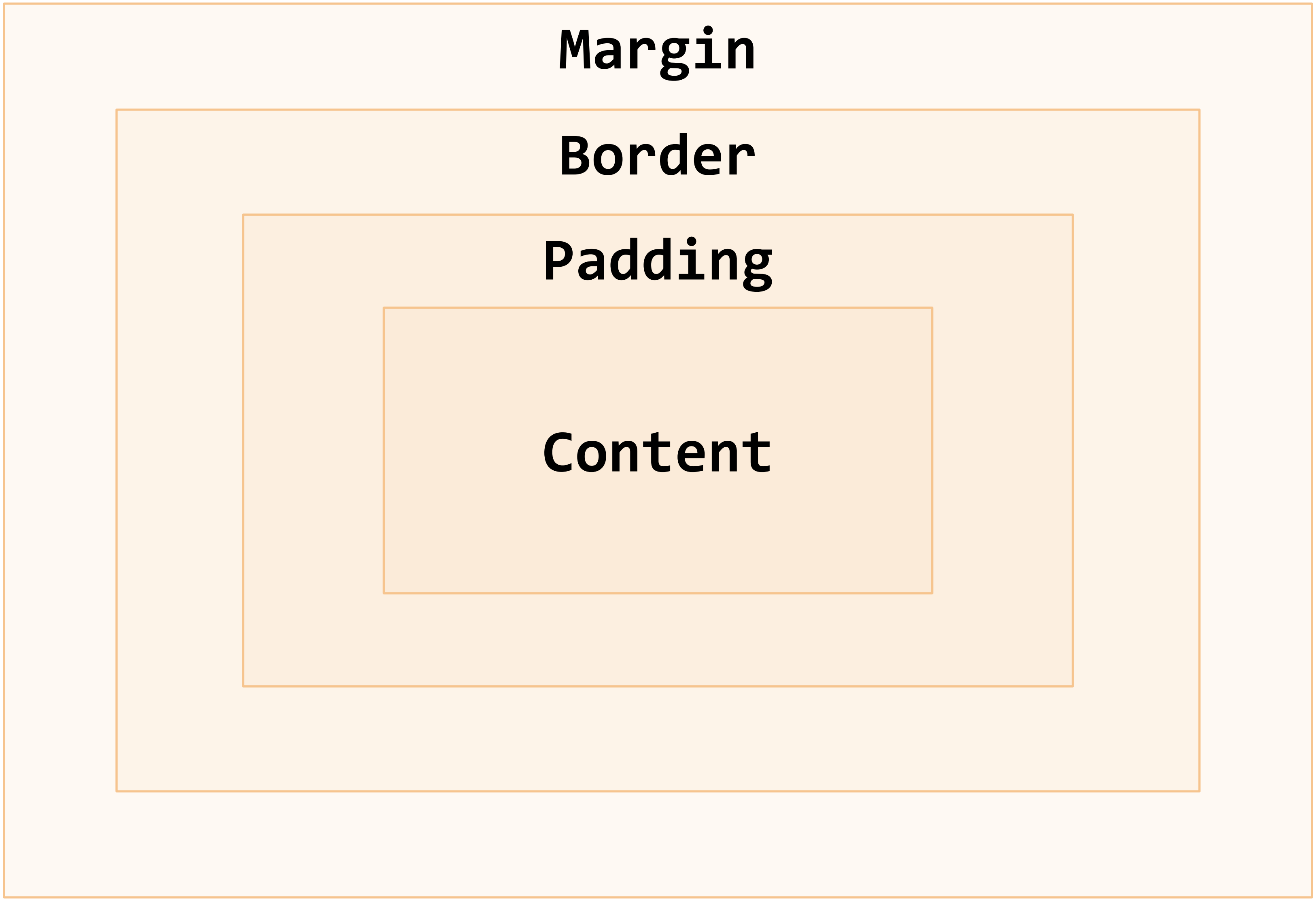
Margins, Borders and Padding

Example: margins-paddings-rules.html

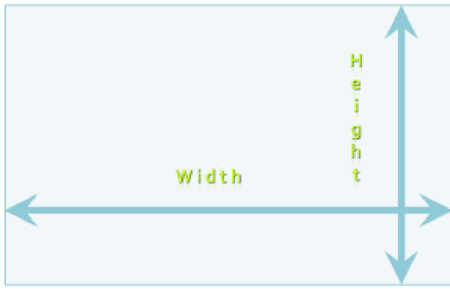
Margins, Borders and Padding

- Understanding the box model
  - A paragraph is a box
  - An image is a box
  - Each tag is a box
  - Boxes within boxes
- **Margin** –the space that separates the boxes
- **Padding** –the space between the border and the contents
- **Border** –the line around each edge of the box

The Box Model



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

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

```
width: 300px;
border: 1px solid black;
padding: 5px;
```

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`

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`

Rounded Corners

- Rounded corners are done by the `border-radius` property

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

Positioning Elements

float-rules.html

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

See example: overflow-rule.html

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
  - floated inline elements can apply height

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

```
:after { content: ""; display:
block; clear: both; height: 0; }
```

Visibility

- `visibility`
  - Determines whether the element is visible
  - `hidden`: element is not rendered, but still occupies place on the page
  - `visible`: element is rendered normally

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)
  - `none`: element is hidden and its dimensions are not used to calculate the surrounding elements rendering (differs from `visibility: hidden`!)

Organize styles / Readability

- DOM Order

```
/* Header */
.header { property:value; }
.header .menu { property:value; }
/* Content */
.content { property:value; }
.content .widget { property:value; }
/* Footer */
.footer { property:value; }
.footer .links { property:value; }
```

- Grouped Order

```
/* Containers */
.header { property:value; }
.content { property:value; }
.footer { property:value; }
/* Navigation */
.header .menu { property:value; }
.footer .links { property:value; }
/* Widgets */
.content .widget { property:value; }
```

Naming / Declaration

Use semantic naming

```
BAD:
.sB {…}
.button3 {…}
.topLeftButton {…}
.greenButton {…}

GOOD:
.searchButton {…}
```

References

- CSS Tutorials

http://www.w3schools.com/css/

- CSS developer guide

https://developer.mozilla.org/en-US/docs/Web/Guide/CSS

- Selectors

- http://code.tutsplus.com/tutorials/the-30-css-selectors-you-must-memorize--net-16048
- http://www.quirksmode.org/css/selectors/