CS 416
Web Programming

Cascading Style Sheets
…just enough to be dangerous

Dr. Williams
Central Connecticut State University

Agenda

- CSS basics
- CSS classes
- Responsive Web Design (RWD)
- CSS frameworks

Cascading Style Sheets

- **C**ascading **S**tyle **S**heets or **CSS**
- Styles define **how to display** HTML elements
- Styles were added to HTML 4.0 **to solve a problem**
- **External Style Sheets** can save a lot of work
- External Style Sheets are stored in **CSS files**

Allow the same page to be styled differently

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

Cascading Style Sheets

- Cascading Style Sheets (CSS) allow you to specify the default style for all elements of a type
- Result is can have consistent formatting throughout page rather than having to respecify for every element
- Allow the HTML file to contain content and CSS contain formatting

CSS Syntax in separate file

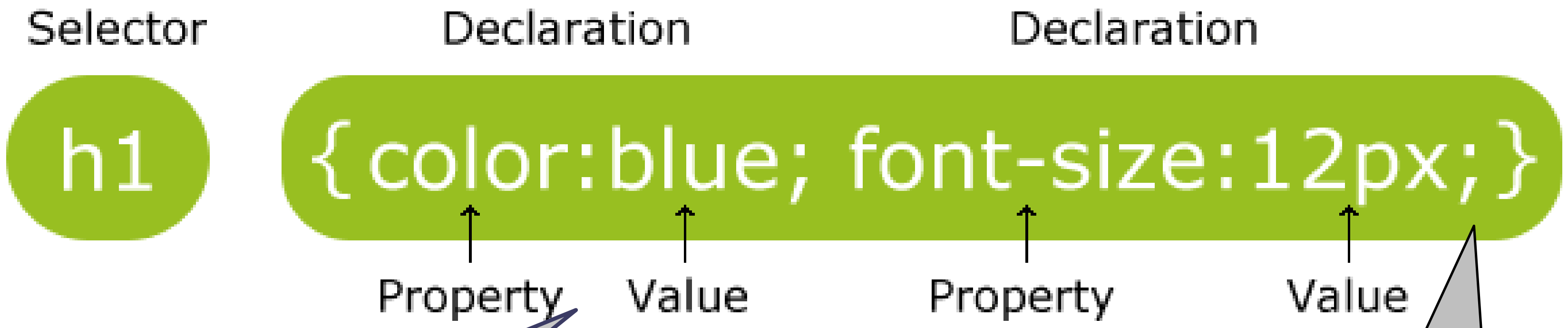
- Specified in <head> by using the <link> tag to specify the page should use an outside file

```
<html>
  <head>
    <link rel="stylesheet" href="ex1.css" />
  </head>
  <body>
```

In this class we will focus on CSS being in a separate file

CSS Syntax

- CSS rules have two main parts:
  - Selector



Syntax of this part is identical to styles within page

Always ends in ";"

CSS syntax cont.

- Each selector matches to the tag with the same name
- Generally to improve readability each style element is put on its own line (although no functional difference)

Tag to apply the style to

```
body {
  background-color:tan;
}
h1 {
  color:maroon;
  /* This is a comment */
  font-size:20pt;
}
```

Common style elements

Property	values
`text-align`	`left, right, center, full`
`text-decoration`	`none, underline, line-through, overline`
`font-style`	`italic`
`font-weight`	`lighter, normal, bold, bolder`
`font-size`	`20px, 200%`
`font-family`	`"Times New Roman", Georgia, Arial, Courier, …`
`color`	`blue, #ff0000, rgb(255,0,0)`

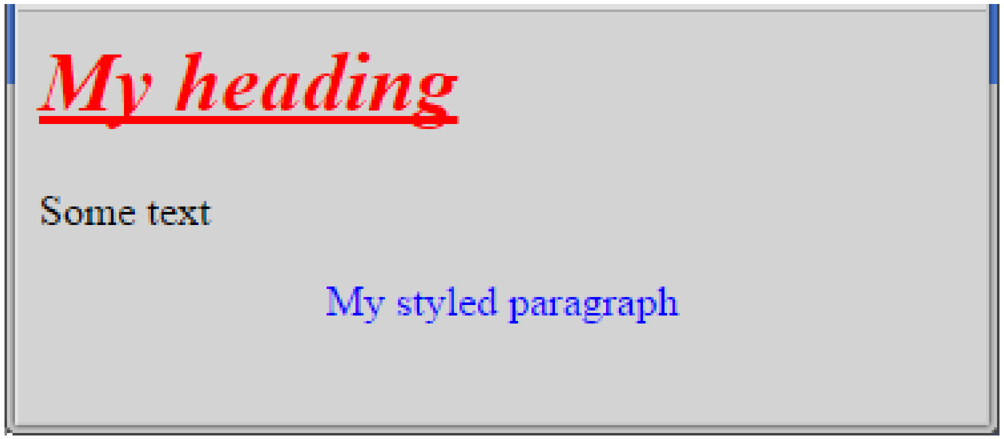
`background`	Can use full syntax like this…
	`background-color: blue;`
	`background-image: url("img_tree.png");`
	`background-repeat: no-repeat;`
	`background-position: right top;`
	`background-attachment: fixed;`
	Or shorthand like this…
	`background: blue url("img_tree.png") no-repeat fixed;`

`list-style-type`	`circle, square, lower-roman, upper-roman, lower-alpha, upper-alpha`
`list-style-image`	`url('sqpurple.gif')`

Simple CSS Example

```html
<html>
 <head>
   <link rel="stylesheet" href="simple.css" />
 </head>
 <body>
  <h1>My heading</h1>
  Some text
  <p>
   My styled paragraph
  </p>
 </body>
</html>
```

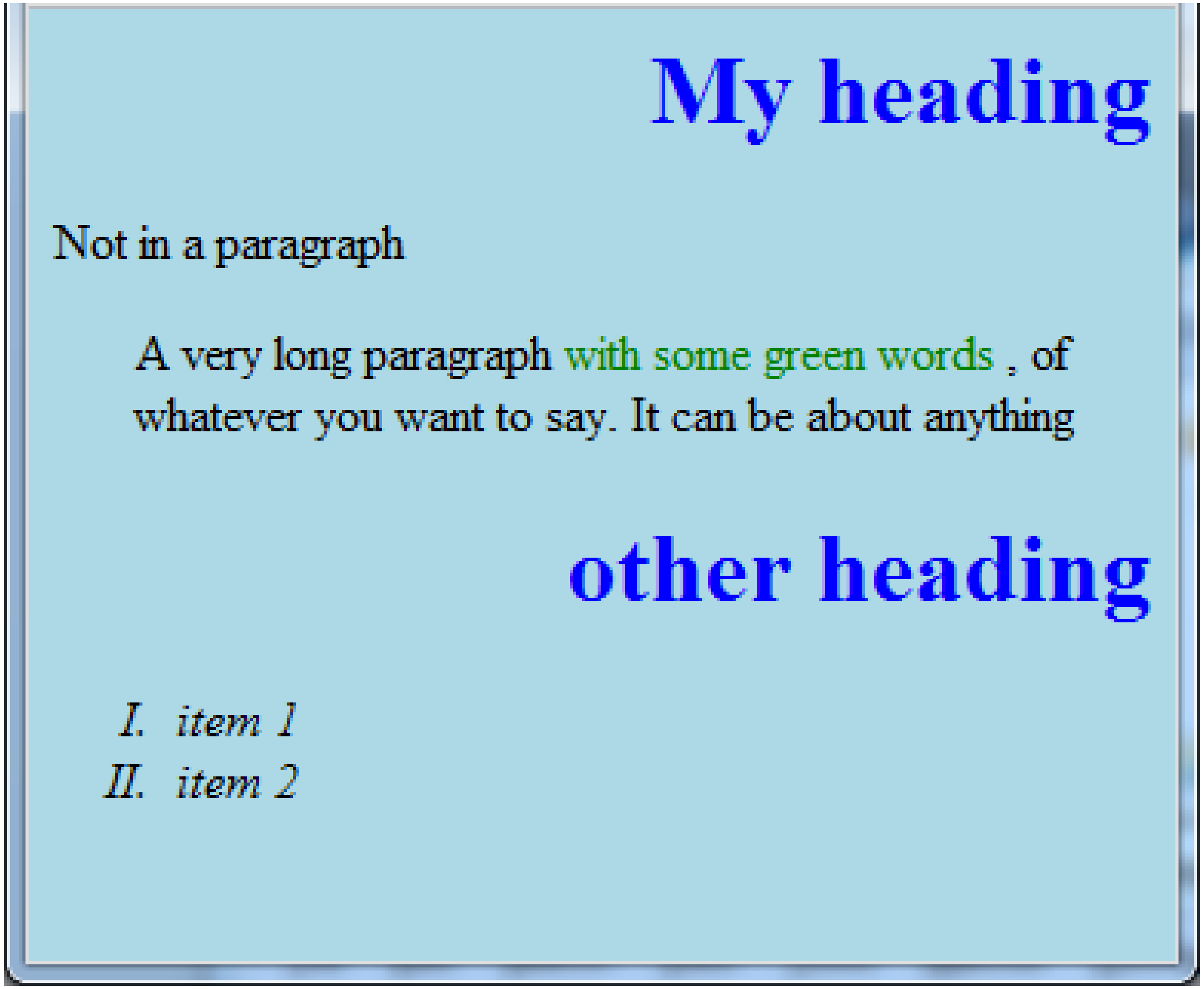
```css
h1{
   color:red;
   font-style:italic;
   text-decoration:underline;
}

p{
   text-align:center;
   color:blue;
}

body{
   background:lightgrey;
}
```


My heading

Some text

My styled paragraph

What would CSS be?



My heading

Not in a paragraph

A very long paragraph with some green words , of whatever you want to say. It can be about anything

other heading

I. item 1
II. item 2

Solution

```
h1{
  color:blue;
  text-align:right;
}

p{
  font-weight:bold;
  text-align:center;
}

ol{
  font-style:italic;
  list-style-type:upper-roman;
}

body{
  background-color:lightblue
}
```

CSS classes

- In addition to defining the format for an entire set of tags you can specify special formatting for a class of tag (a subset of that tag)
- For example you may want all paragraph tags to be blue, but you want important paragraphs to also be in italics

CSS classes continued

- To specify this special case you use a “.” to indicate a special **class** of the formatting

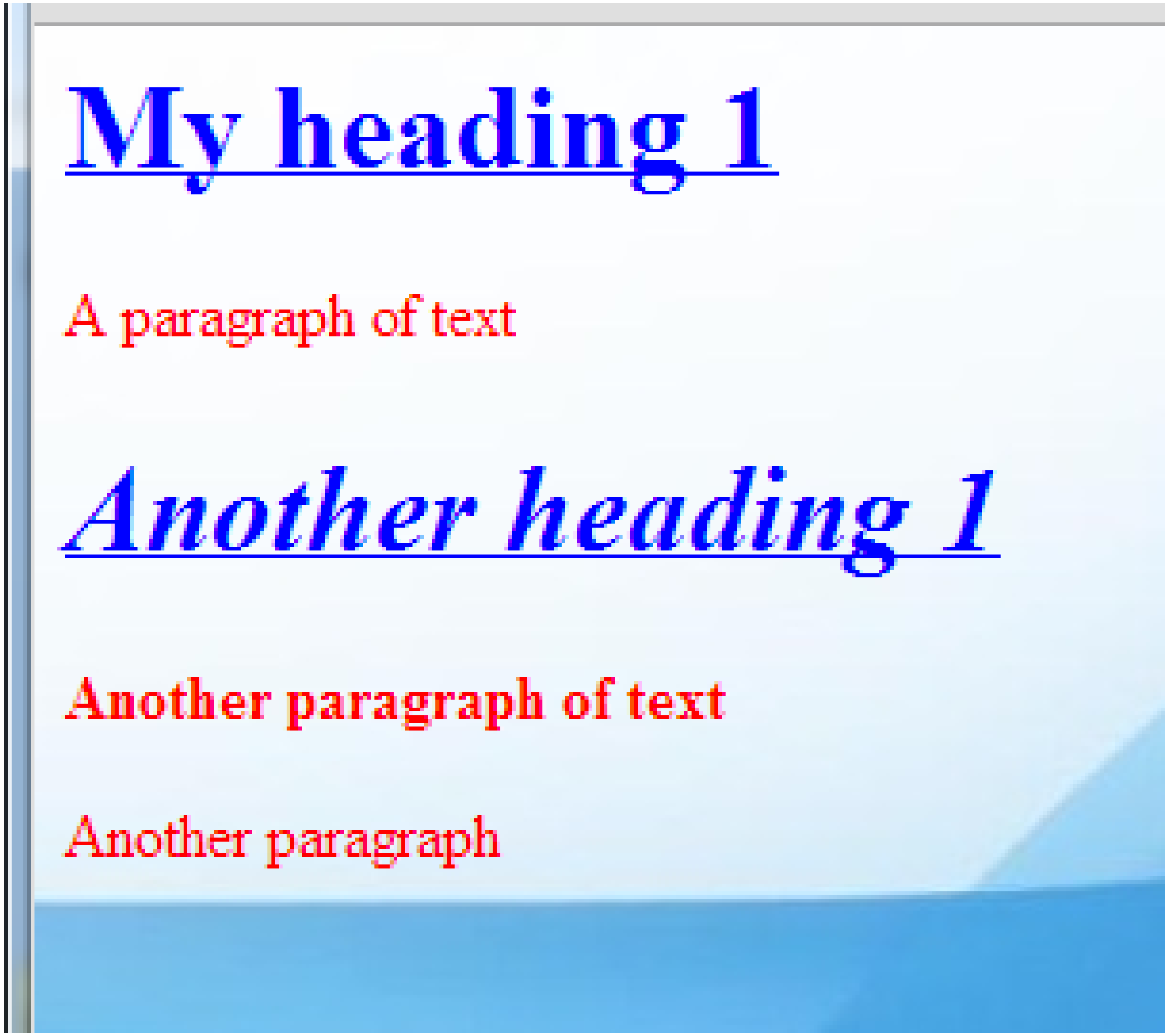
```
p{color:blue}
p.important{font-style:italic}
```

- To reference this specific class of style in your html you would add class=“important” where whatever is in quotes is the class style to use

```
<p>some normal paragraph</p>
<p class="important">important paragraph</p>
```

- In this case the normal paragraph style will be applied as well as the class paragraph style so the text will be both blue and italics

What might the HTML and CSS look like?



My heading 1

A paragraph of text

Another heading 1

Another paragraph of text

Another paragraph

Classes continued

- In addition to being able to specify a style for a specific tag (ex. p.important) you can also specify a general style for all tags by preceding the style with a "."

```
.center{text-align:center}
```

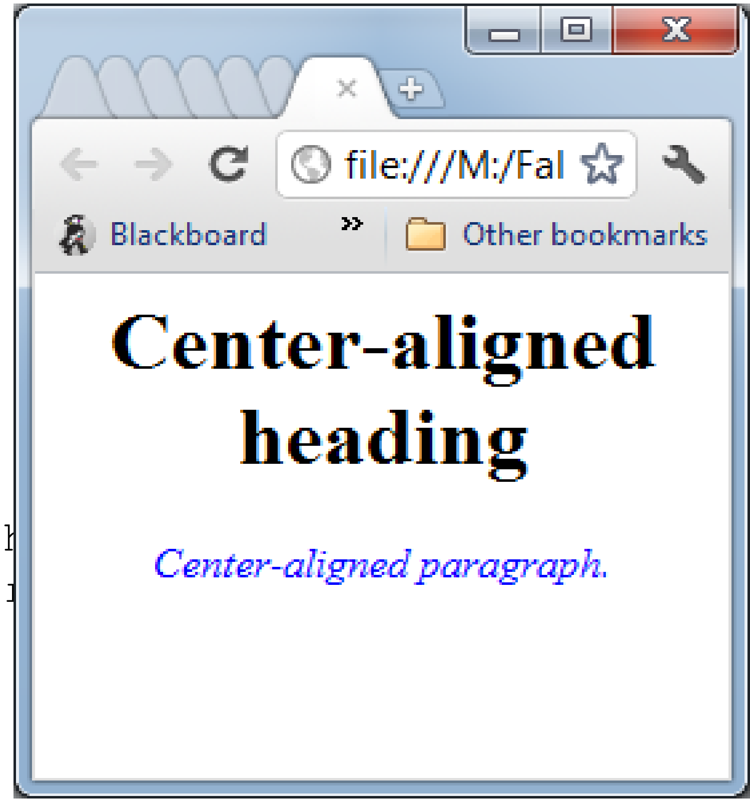
Then in the HTML

```
<h1 class="center">my heading</h1>
<p class="important center">my
paragraph</p>
```

- Multiple classes can be specified by separating classes with a space

Multiple classes

```
p{
  color:blue
}
p.important{
  font-style:italic;
}
.centered{
  text-align:center;
}

<body>
<h1 class="centered">Center-aligned heading</h
<p class="important centered">Here is an impor
  paragraph that is also centered.</p>
</body>
</html>
```



Center-aligned heading

Center-aligned paragraph.

The style elements are additive. If one overrides another the more specific one is what is used (ex. style of a class of a paragraph overrides style that is applied to all paragraphs in general)

Many, many more options…

- Select by:
  - id `#myid`
  - attribute presence `[myattr]`
  - attribute value `[myattr="myvalue"]`
  - position in hierarchy (heading inside a table tag)
  - Current interaction as well (pseudo classes):
    - Mouse hover `h1:hover`
    - Focus
    - Link visited/unvisited

See demo: http://www.w3schools.com/cssref/trysel.asp

Put simply if there is a will there probably is a way

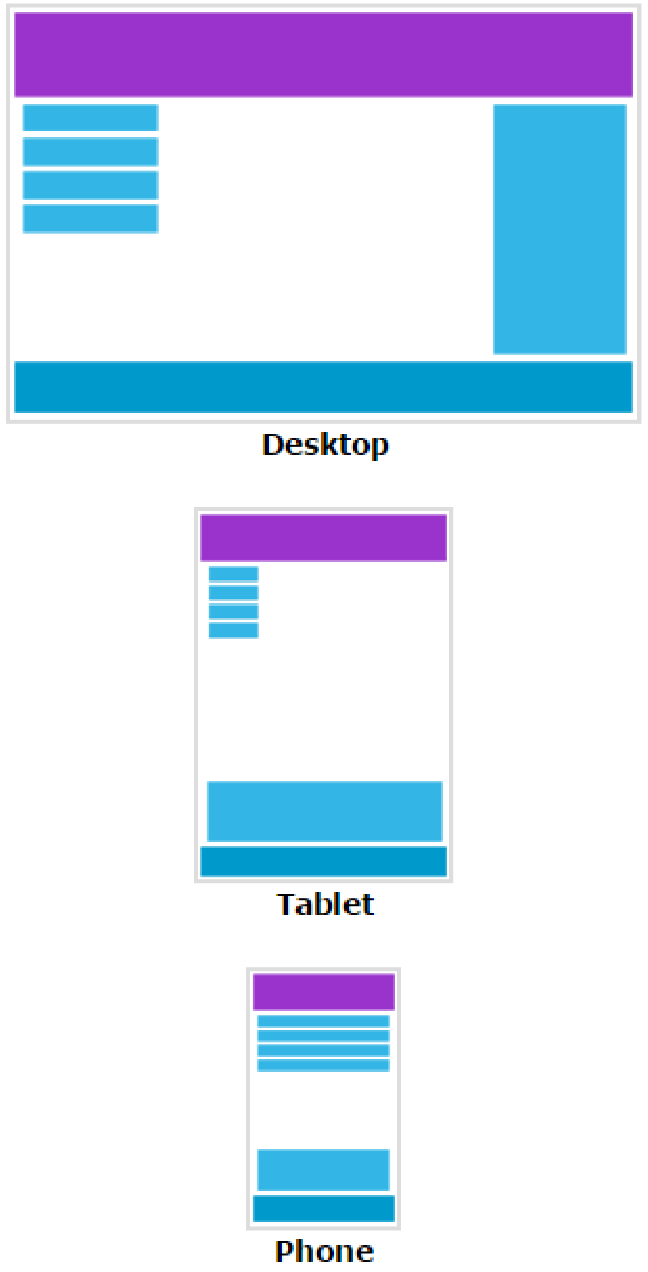
CSS Responsive Web Design (RWD)

- Adapt view to be best suited for size of display: desktop, tablet, phone, ???
- Responsive web design uses only HTML and CSS.
- Responsive web design is not a program or a JavaScript.

Principle of RWD

- If screen is smaller don’t leave out information, but rather adapt how information is displayed

- Big change is rather than having sizes absolute, scale to device size AND adjust layout of elements



RWD - Viewport

- Principle is use viewport is the user's visible area of web page – acceptable to scroll vertically, avoid need for horizontal scrolling
- Set the viewport in head of document

```
<meta name="viewport"
      content="width=device-width,
      initial-scale=1.0">
```

- Size content to viewport using relative sizing i.e. 100% width…but be careful

Different layouts

- Have CSS such that different layouts used if threshold reached

```
/* For mobile phones: */
[class*="col-"] {
    width: 100%;
}
@media only screen and (min-width: 768px) {
    /* For desktop: */
    .col-1 {width: 8.33%;}
    .col-2 {width: 16.66%;}
    .col-3 {width: 25%;}
    .col-4 {width: 33.33%;}
    ...
```

There is so much to learn!

- Bottom line is there is a ton of cool look and feel you can customize and strategic layout using RWD by getting into the nitty gritty of CSS

- However…you could easily spend countless working through and debugging CSS particularly related to RWD

- Enter CSS frameworks!

CSS Frameworks

- Provide sets of common functionality
  - Many built in commonly grouped styles
  - Commonly accepted look and feel
  - RWD – layouts and automatic adjustment at screen size cutoffs to common layout reorganizations
- Most popular ones (http://www.vermilion.com/responsive-comparison/)
  - Bootstrap – HTML, CSS, JS framework w/Sass
  - Foundation - HTML, CSS, JS framework w/Sass
  - Pure – HTML, CSS, extremely small
  - W3.css
  - Many others available