Welcome

What This Course Will Cover

- Cascading Style Sheets
  - Why they are a better way
  - The syntax
  - Development Tools
  - Accessibility Issues

Week One

- Focus is on getting your feet wet:
  - What are the three common methods for styling your code?
  - How do these methods interact?
  - Basic styling of fonts
  - Introduction to placing elements

Week Two

- The Box Model
- Styling links and lists
- Advanced Selectors
- Browser Capabilities

Week Three

- Pseudo Classes and Elements
- Transitions
- Transforms
- Positioning

Week Four

- Samples, samples, samples
  - Styling a table
  - Styling a navigation menu
- Working on Final Project

Who is this class for?

- This class is for those who are already familiar with HTML5.
- Who have persistence
- Who are willing to do more than I do in these lectures.

Who Am I?

- Ph.D. in Computer Science
- Two decades of teaching experience
- Emphasis on education for those who running around classrooms while helping students debug

Workload

- Weekly quizzes – short
- Weekly peer-graded assignments
    - Demonstrate general capabilities to code what we have learn. Every student will have the same html, but will create a unique look.

Succeeding in This Class

- In a perfect world you would code with a friend…so use the message boards.
- Never spend more than 15 minutes on one thing that doesn’t work. Move on.
- Look things up on your own!
- Practice, practice, practice!

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

CSS3
Cascading Style Sheets

Adding Style to your Pages

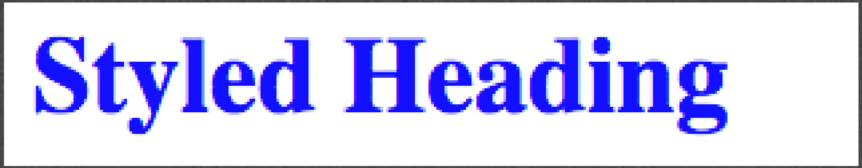
Browser Default Styling

- The same html file may look different when viewed on different browsers.
    - Some tags are supported, some aren't
    - Browsers may have different *default styles*
- In general, default looks are plain.

Adding Style

- As styling tags were phased out of html, styling was done with style attribute

```
<h1 style = "color:blue">Styled Heading</h1>
```

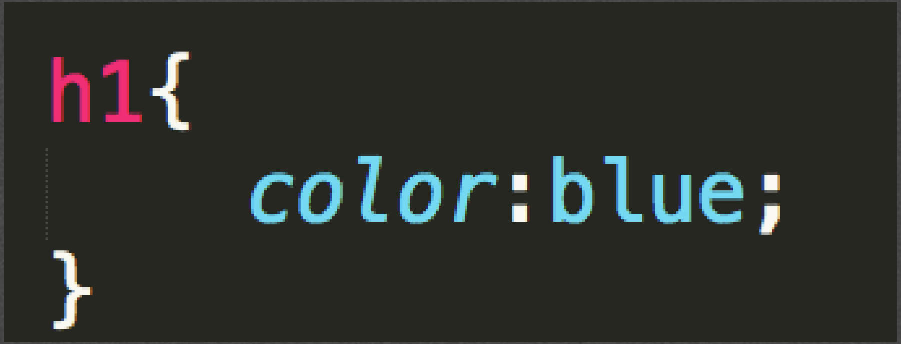

Styled Heading

- Violated separation of content/style

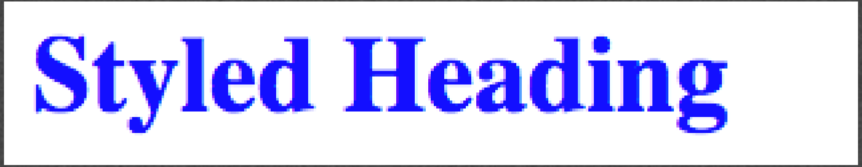
Cascading Style Sheet

- CSS defined generic rules that can apply to multiple elements

```
selector {
        property: value;
}
```



```
h1{
    color:blue;
}
```



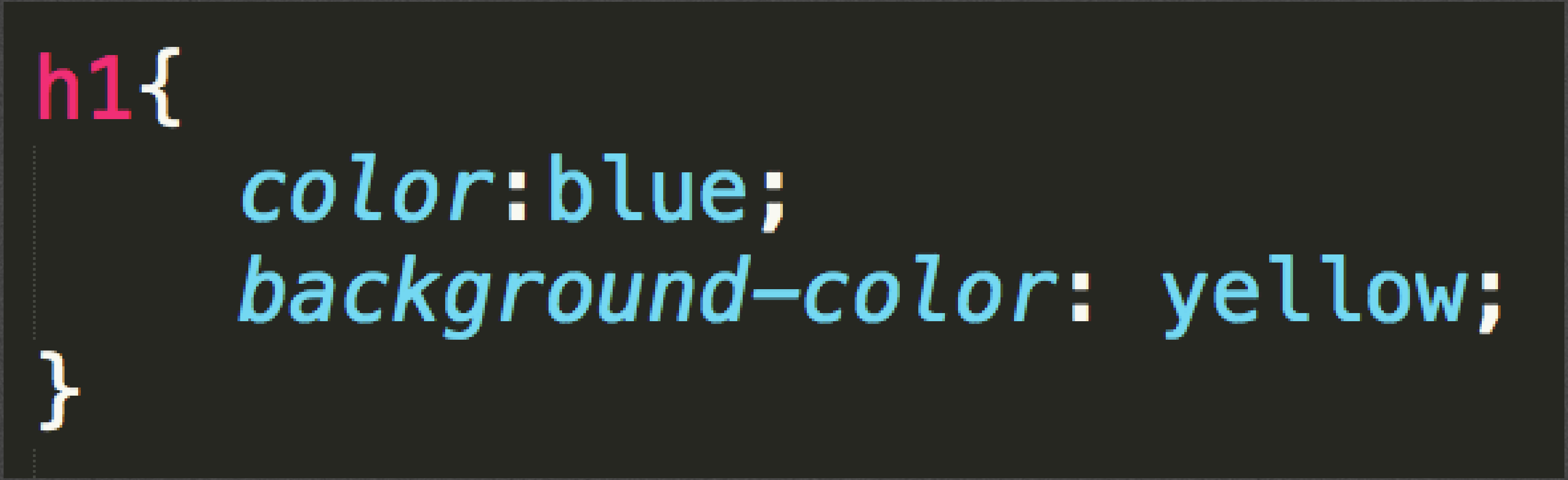
Styled Heading

Rule Syntax

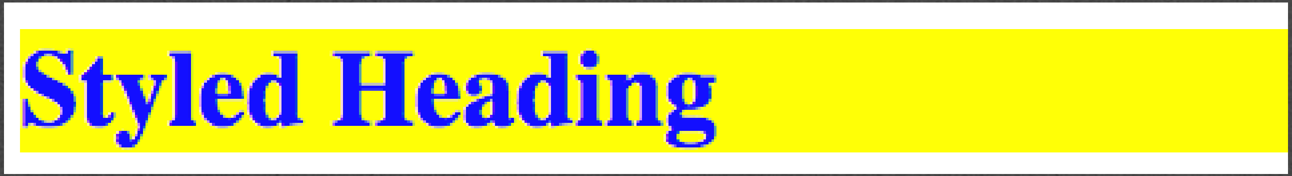
- Brackets and semicolons are very important
- This is where a good editor can make a BIG difference

```
/* This is how comments are done */
```

Multiple Properties



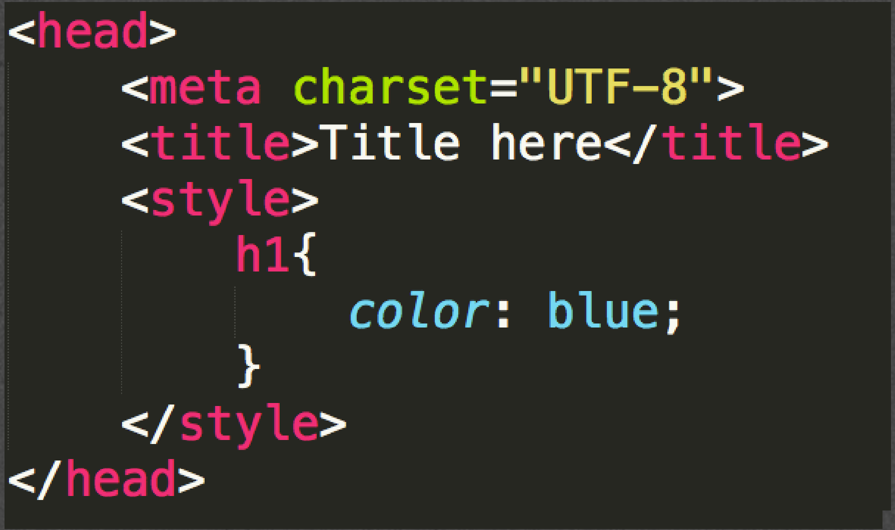
```
h1{
        color:blue;
        background-color: yellow;
}
```



Styled Heading

Internal Style Sheet

- Styling is defined within <head>
- Rules are defined within <style>
- Styles are applied to all elements in that file



```
<head>
    <meta charset="UTF-8">
    <title>Title here</title>
    <style>
        h1{
            color: blue;
        }
    </style>
</head>
```

- Don’t forget to close the style tag!!

External Style Sheet

- You can put rules in an external file (don’t use the style tag!!)
- A link to the style sheet is put in the head section.

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

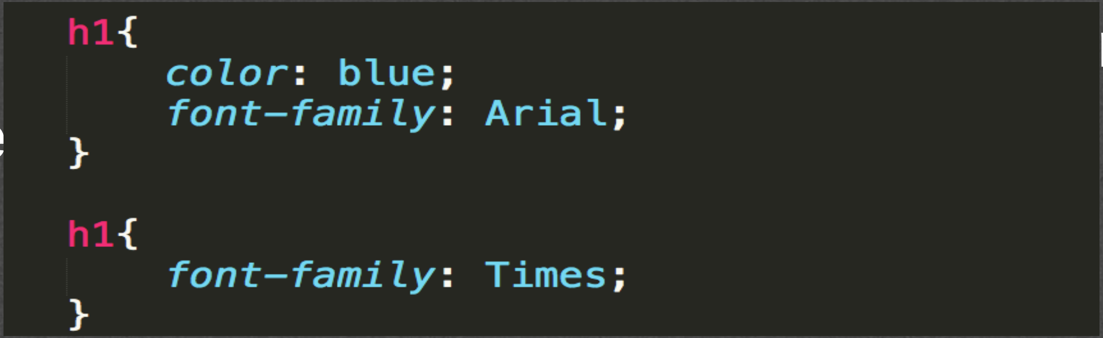
- Styles are applied to all elements in all files that links to the style sheet

The “Cascading” part of CSS

- Browser default
- *External* style sheets
- *Internal* style (in the head section)
- *Inline* style (inside an HTML element)

Rule precedence

- What if one selector is defined in two external files?
    - The rules from the most recent file have precedence
- What if o n the same file



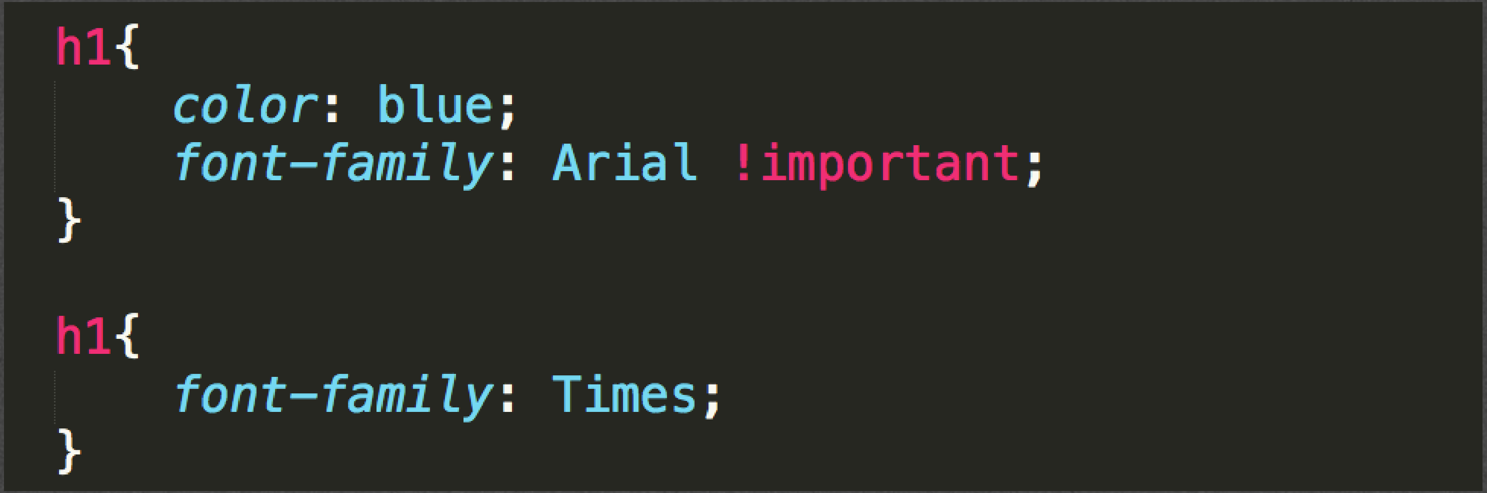
```
h1{
      color: blue;
      font-family: Arial;
}

h1{
      font-family: Times;
}
```

    - The most recent rule has precedence

!important

- It is possible to override later rules, use !important



```css
h1{
	color: blue;
	font-family: Arial !important;
}

h1{
	font-family: Times;
}
```

Example

Review

- Why do we want/need to separate content from formatting?
- How does this also tie in to external/internal style sheets?
- Understand that this is very powerful. See http://www.csszengarden.com/

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

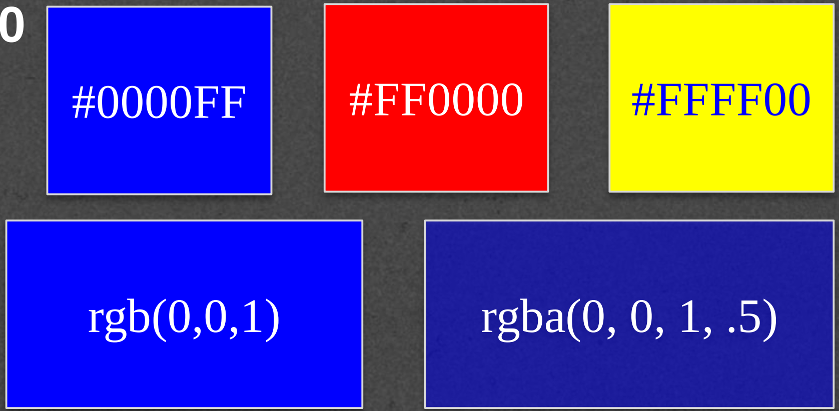
BY NC

Colors

Using the best colors for your site

Color Conventions

- Color names (blue, red, yellow, etc.) work, but should be avoided
- Hexadecimal is common convention
  - #0000FF, #FF0000, #FFFF00
- rgb
  - (0, 0, 1), (1, 0, 0), (1, 1, 0)
- rgba
  - (0, 0, 1, .5)



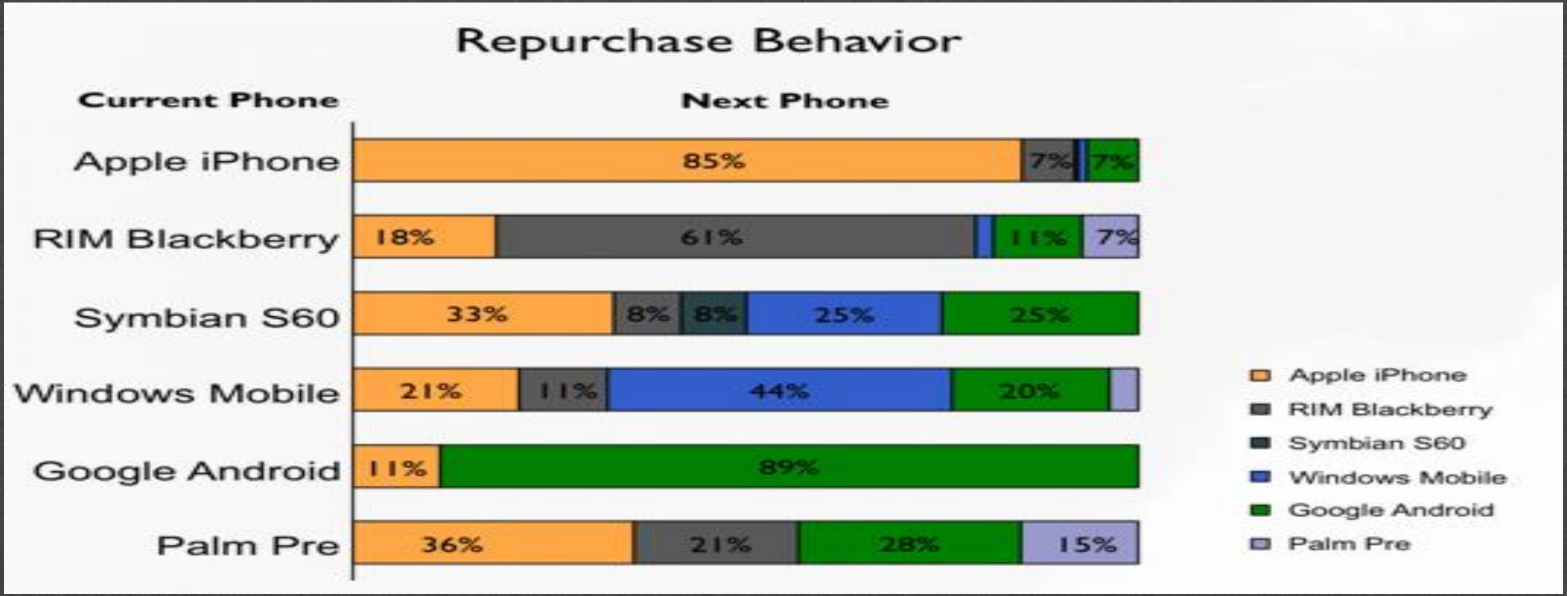
Accessibility

- Appropriate use of color is critical to web accessibility
- Many more people are visually impaired or color blind than are legally blind

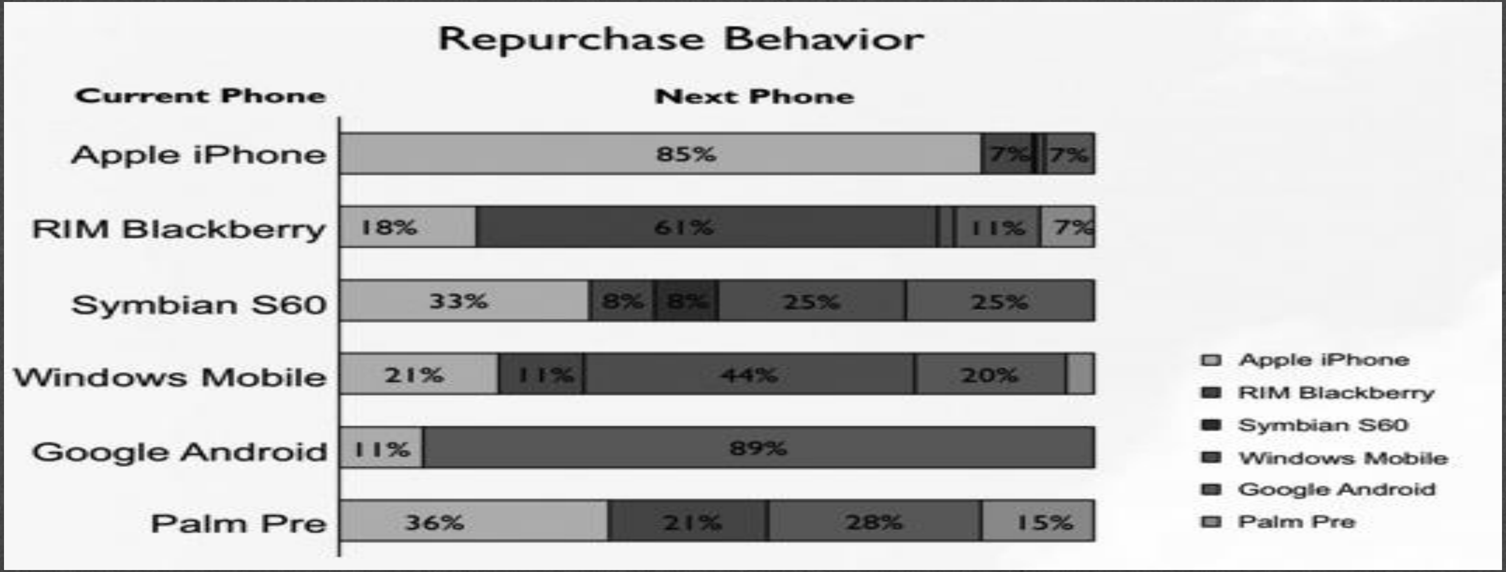
What is color contrast?

- You intuitively know when something has poor contrast
- There are tools that quantify the contrast between text and its background
- http://wave.webaim.org/
- http://webaim.org/resources/contrastchecker/

Don’t use color alone to convey meaning



Test in gray scale …



Review

- Use web safe colors and use an accepted convention
- Test your site using a contrast checker
- Avoid using color to convey meaning

Acknowledgements/Contributions

Styling Your Text

Styling your text

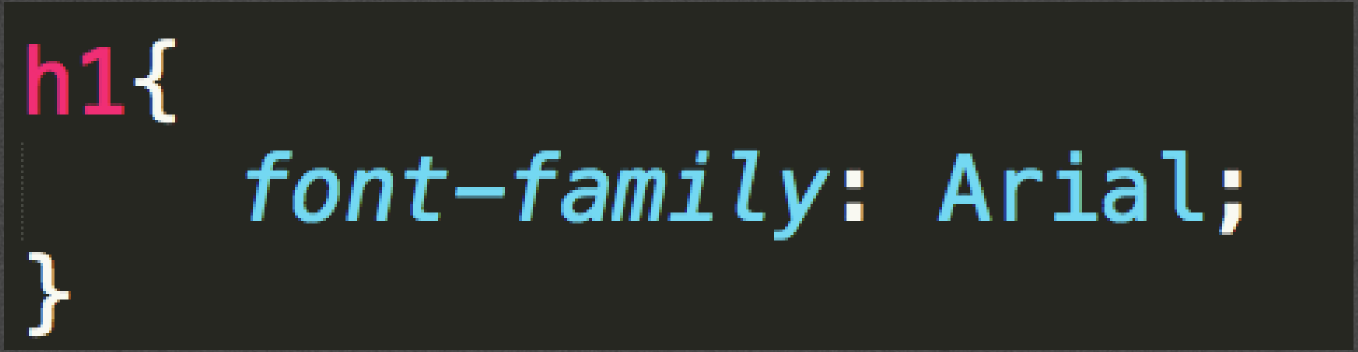
Options

- Many options for styling your text:
  - font (family, style, variant, size)
  - color and background
  - alignment
  - line-height

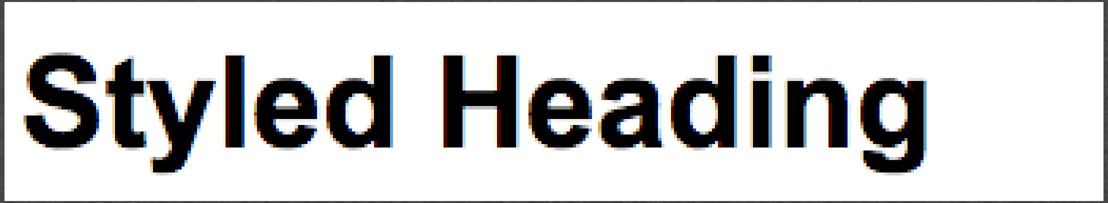
font-family

- **Font families are styles of text**
- **Examples:**
  - Helvetica, `Courier`, “`Courier New`”, “Comic Sans MS”, *cursive*, Verdana

font-family



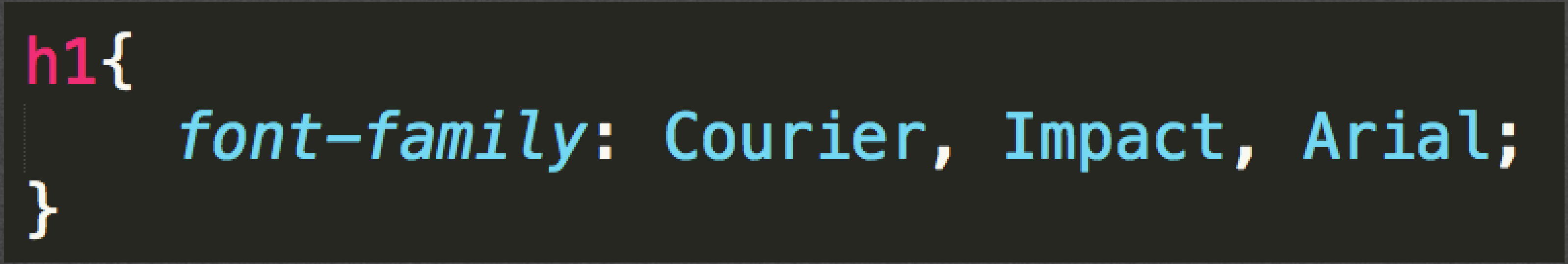
```
h1{
    font-family: Arial;
}
```



Styled Heading

font-family

- Not all font-families supported by all of the operating systems, so you can provide alternatives.



```
h1{
    font-family: Courier, Impact, Arial;
}
```

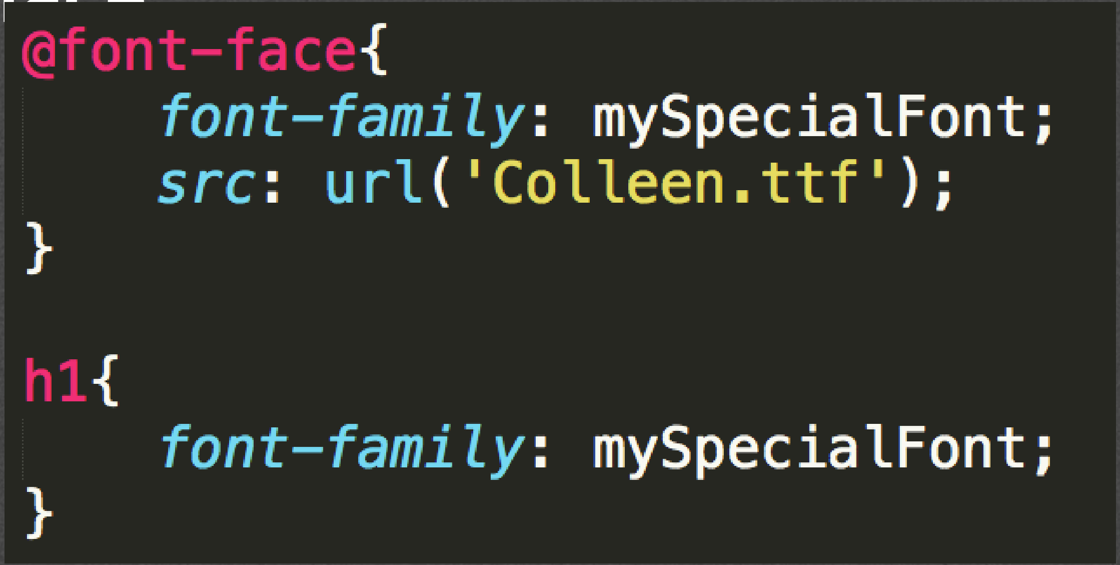
font-family Considerations

- Some fonts are not as user-friendly, use sans-serif when possible.

Test Test

Custom fonts

- To expand beyond "web-safe" fonts use @font-face

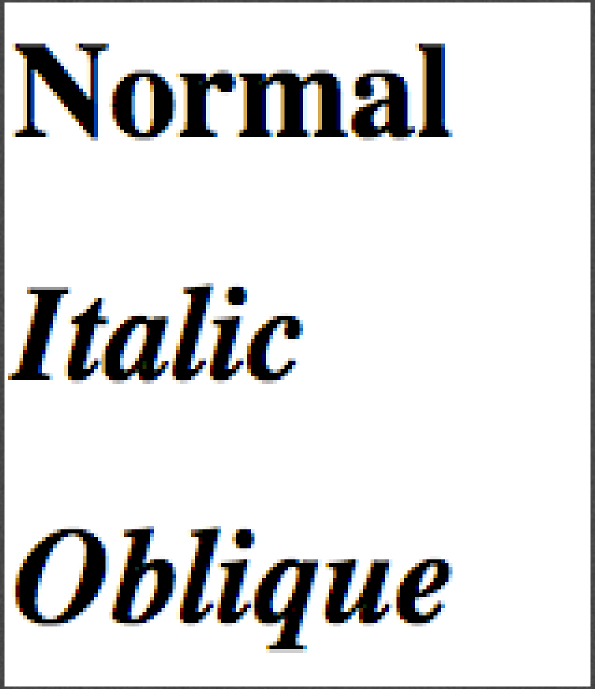


```
@font-face{
	font-family: mySpecialFont;
	src: url('Colleen.ttf');
}

h1{
	font-family: mySpecialFont;
}
```

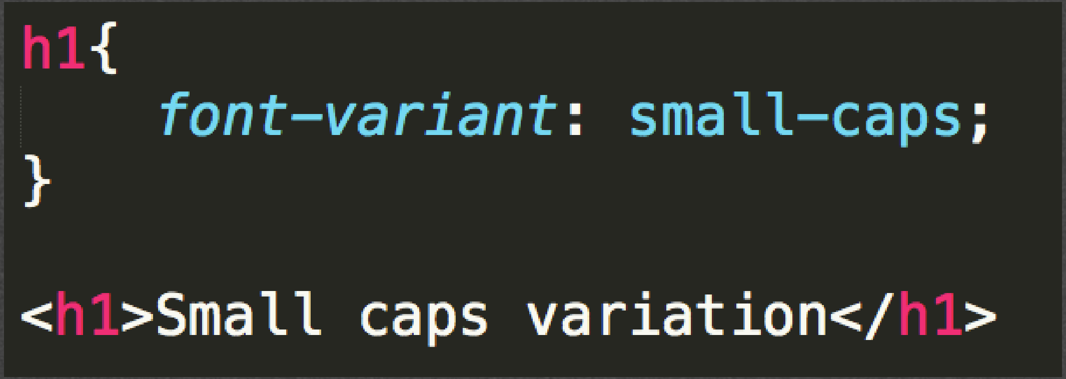
font-style

- font-style:
  - normal
  - italic
  - oblique


Normal
Italic
Oblique

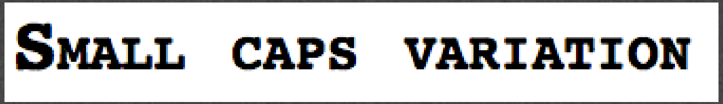
font-variant

- font-variant:
  - normal
  - small-caps



```
h1{
    font-variant: small-caps;
}

<h1>Small caps variation</h1>
```



Small caps variation

font-size

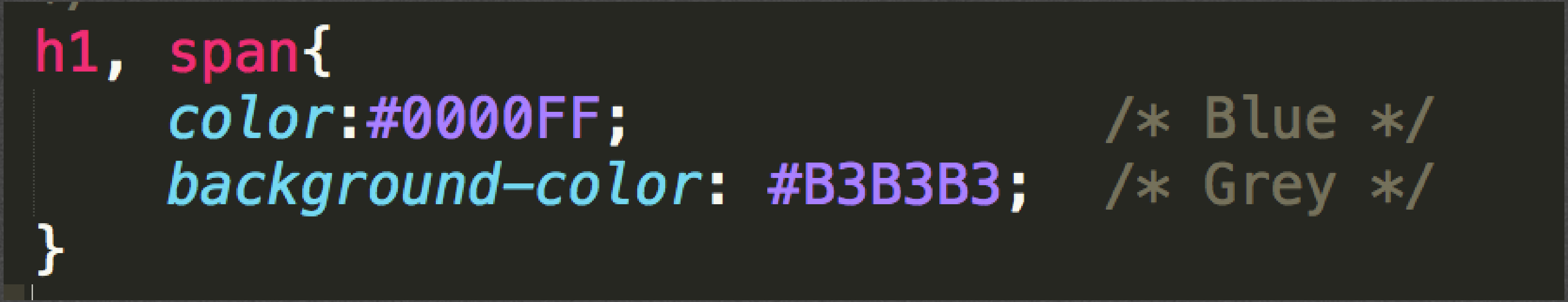
- This is only the beginning of our discussion on sizes…
- Options
  - xx-small, x-small, small, smaller
  - medium
  - larger, x-large, xx-large, larger
  - Use pixel
  - Use %

color and background-color

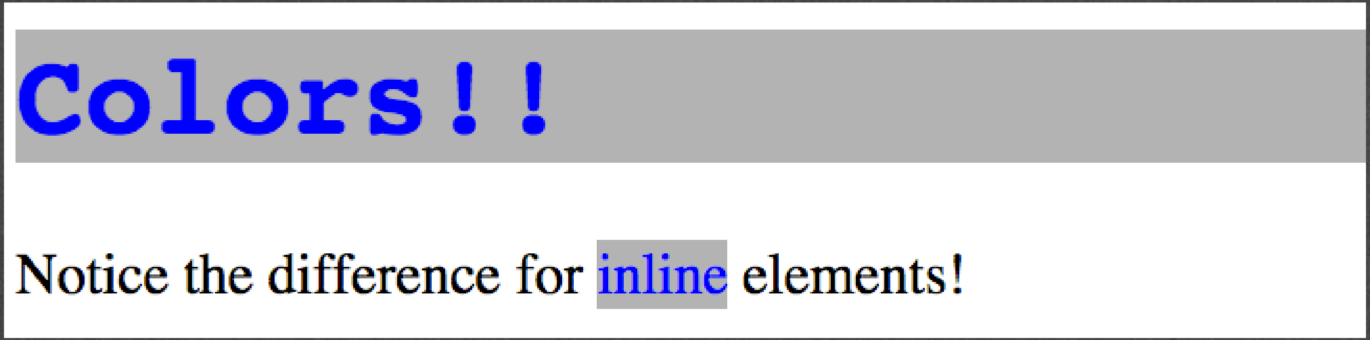
- The color attribute is the color of the foreground.

- The background-color is the color of the background

colors



```
h1, span{
    color:#0000FF;                  /* Blue */
    background-color: #B3B3B3;      /* Grey */
}
```



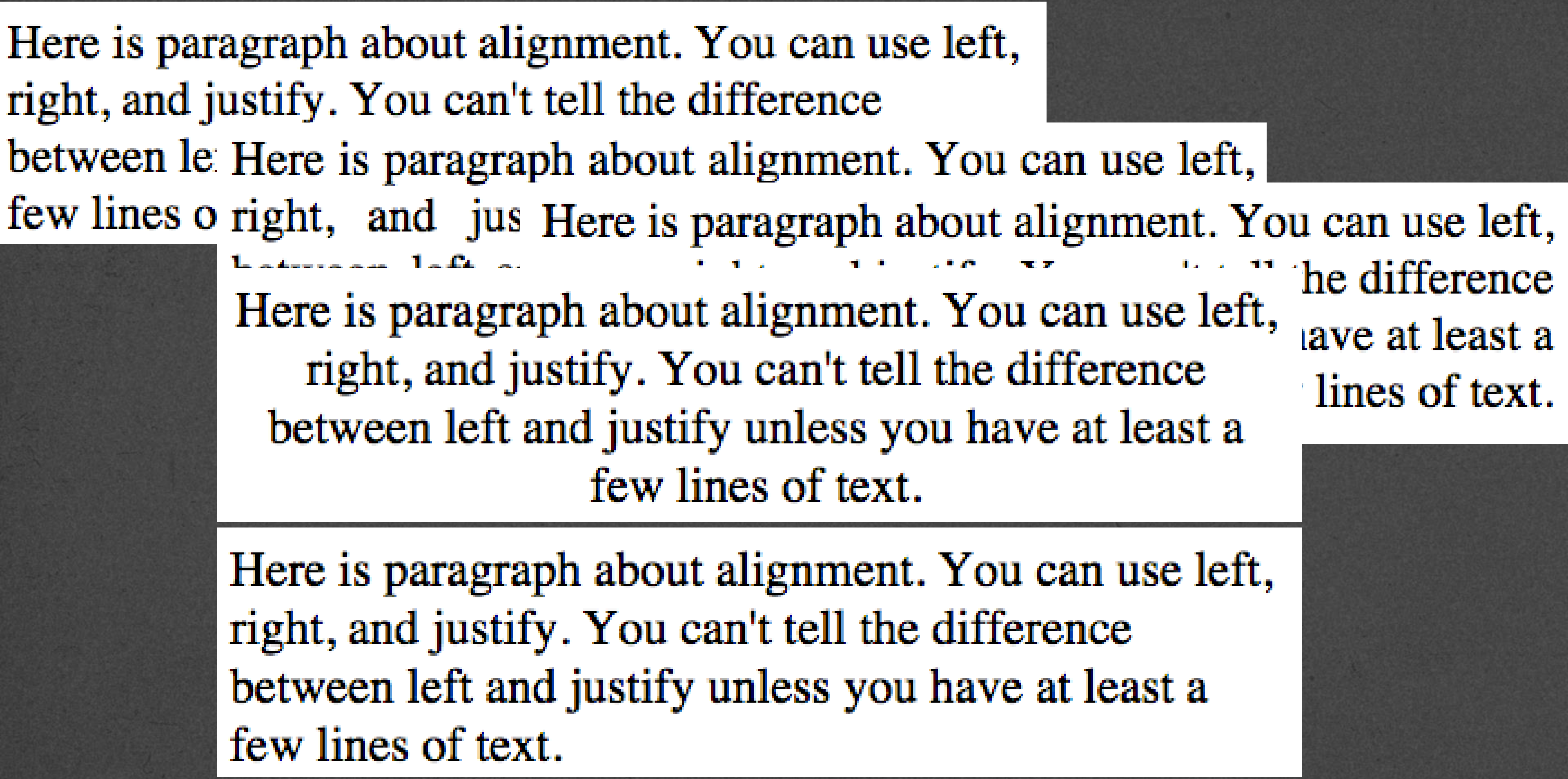
Colors!!

Notice the difference for inline elements!

text-align

- Aligning text is simple!
- text-align
    - left
    - right
    - center
    - justify

01.01 Welcome



Here is paragraph about alignment. You can use left,
right, and justify. You can't tell the difference
between le
few lines o

Here is paragraph about alignment. You can use left,
right, and jus

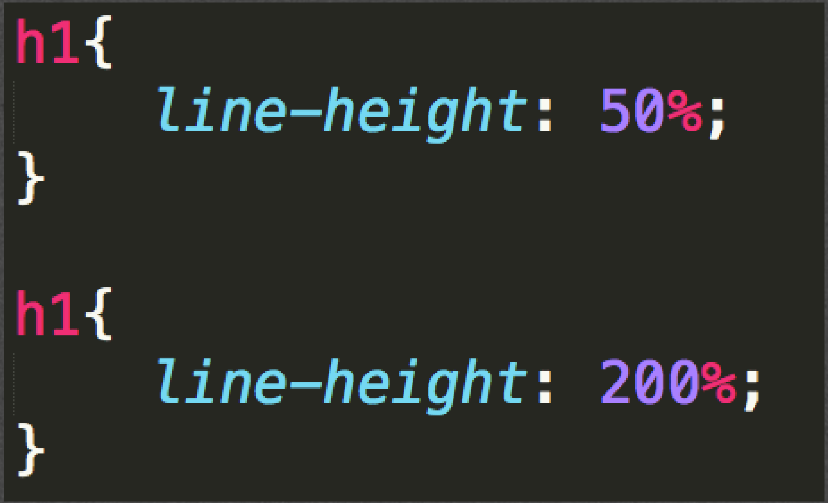
Here is paragraph about alignment. You can use left,
he difference
ave at least a
lines of text.

Here is paragraph about alignment. You can use left,
right, and justify. You can't tell the difference
between left and justify unless you have at least a
few lines of text.

Here is paragraph about alignment. You can use left,
right, and justify. You can't tell the difference
between left and justify unless you have at least a
few lines of text.

line-height

- As you can guess, doesn't affect font
- Adjusts the space between the lines of text



```
h1{
	line-height: 50%;
}

h1{
	line-height: 200%;
}
```

Review

- The number of options for styling text can seem overwhelming.
- Practice on toy problems!
- Design larger projects on paper first!!!

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

Display and Visibility

Display is Key to Layout

- Every element is a box
- Display affects the layout of neighboring elements

Common Values

- inline: sits next to other elements
  - takes up “just enough” width and height
- block: forces line break
  - default: take up all horizontal width and “just enough” height
  - rules can set height and width

Common Values

- inline-block:
  - same as inline, but accepts height and width
- none: removed from page
  - Still in DOM, but not visual (even to SRs)

Example

Complementary Properties

- float
  - Reposition elements to the right or left.
  - Elements are aware of one another and will not overlap.
  - Values: left, right
- clear
  - Used to keep floating elements away
  - Values: left, right, both

Example

Element Overflow

- What happens when you set a height/width and the content doesn’t fit any longer?
- Use overflow to determine access

Overflow

- visible: Can cause text to show up "on top" of other text
- hidden: Hides anything that goes beyond bounding box
  - This can cause problems since if the user increases font size, they may not be able to see content
- scroll: Gives horizontal and vertical scrollbars
- auto: Adds scrollbars as needed

Example

Other Display Values

- New display properties are available, but not always supported:
  - Table
  - Grid
  - Flexbox

display:table

- Sometimes you want to have table-like layout without using table structure, use display:table along with display:table-cell for elements.

Example

Visibility

- Specifies whether or not element is visible
- Options include:
  - visible
  - hidden
  - collapse (only for table elements)
- Unlike display:none a hidden element is still part of the DOM and still takes up space

Review

- Display is just one tool for positioning our elements on the page
- Early design will make the coding easier
- Utilize tools to see the different options

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

BY NC

Homework One

Adding style with an external Style Sheet

Objective

- Create your own unique style sheet that will be used by three different html files.

Before

http://www.intro-webdesign.com/CSS/assignment-1/index.html

http://www.intro-webdesign.com/CSS/assignment-1/teams.html

http://www.intro-webdesign.com/CSS/assignment-1/history.html

After

http://www.intro-webdesign.com/CSS/assignment-1/index.jpg

http://www.intro-webdesign.com/CSS/assignment-1/teams.jpg

http://www.intro-webdesign.com/CSS/assignment-1/history.jpg

You must leave the HTML alone

Create style sheet that updates:

- header, headings, section, links and images

header

- Change background color

h1 and h2

- Change the font color
- Make sure the heading is displayed in all capital letters
- For just h1:
    - Center the text

section

- Change the background color

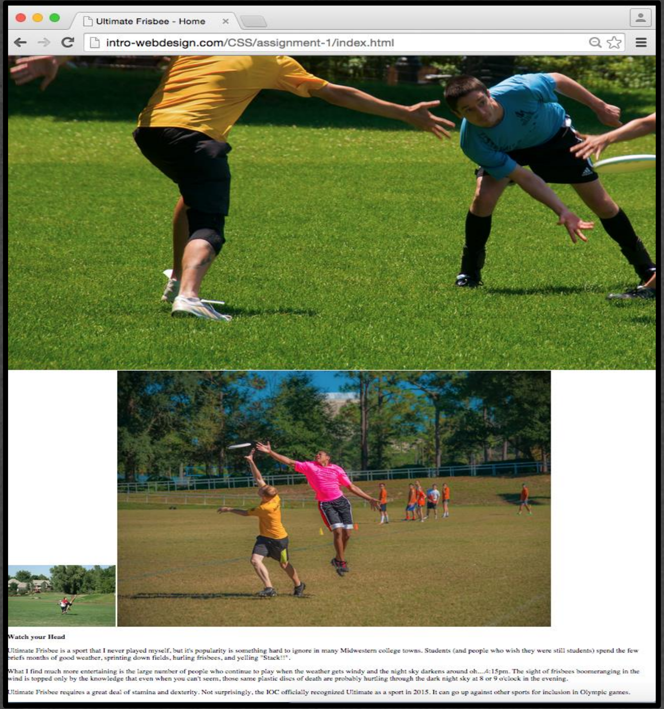
Links

- Change the background color and font color
- Center the text (subtle change)

Images

- Since we haven’t talked yet about the best way to get those images where we want them, just use CSS3 to remove them from the page.

BEFORE



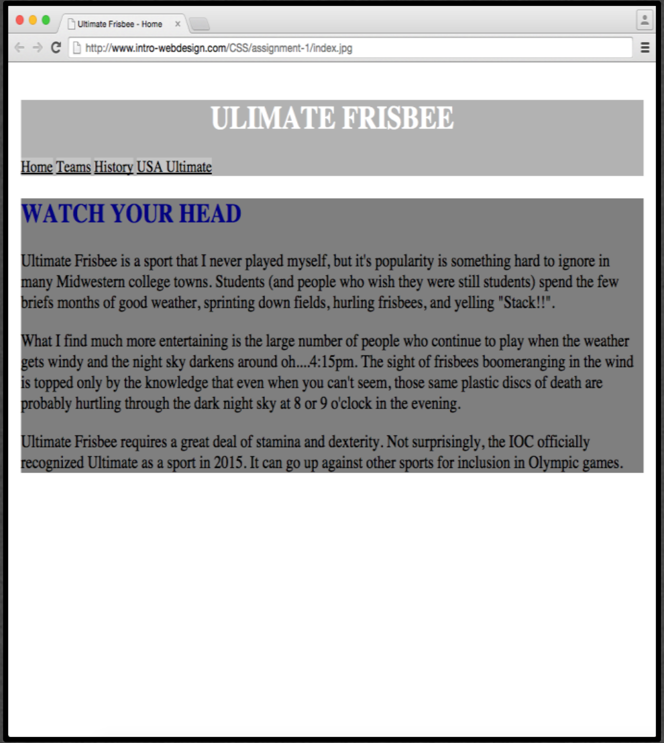
Watch your Head

Ultimate Frisbee is a sport that I never played myself, but it's popularity is something hard to ignore in many Midwestern college towns. Students (and people who wish they were still students) spend the few briefs months of good weather, sprinting down fields, hurling frisbees, and yelling "Stack!!".

What I find much more entertaining is the large number of people who continue to play when the weather gets windy and the night sky darkens around oh....4:15pm. The sight of frisbees boomeranging in the wind is topped only by the knowledge that even when you can't seem, those same plastic discs of death are probably hurtling through the dark night sky at 8 or 9 o'clock in the evening.

Ultimate Frisbee requires a great deal of stamina and dexterity. Not surprisingly, the IOC officially recognized Ultimate as a sport in 2015. It can go up against other sports for inclusion in Olympic games.

AFTER



ULIMATE FRISBEE

Home Teams History USA Ultimate

WATCH YOUR HEAD

Ultimate Frisbee is a sport that I never played myself, but it's popularity is something hard to ignore in many Midwestern college towns. Students (and people who wish they were still students) spend the few briefs months of good weather, sprinting down fields, hurling frisbees, and yelling "Stack!!".

What I find much more entertaining is the large number of people who continue to play when the weather gets windy and the night sky darkens around oh....4:15pm. The sight of frisbees boomeranging in the wind is topped only by the knowledge that even when you can't seem, those same plastic discs of death are probably hurtling through the dark night sky at 8 or 9 o'clock in the evening.

Ultimate Frisbee requires a great deal of stamina and dexterity. Not surprisingly, the IOC officially recognized Ultimate as a sport in 2015. It can go up against other sports for inclusion in Olympic games.

Peer grading

- Grades will be based on level of completion, not aesthetics
- Proper standards do apply
- Only style the listed elements, otherwise it makes it harder to grade

Acknowledgements/Contributions

Box Model

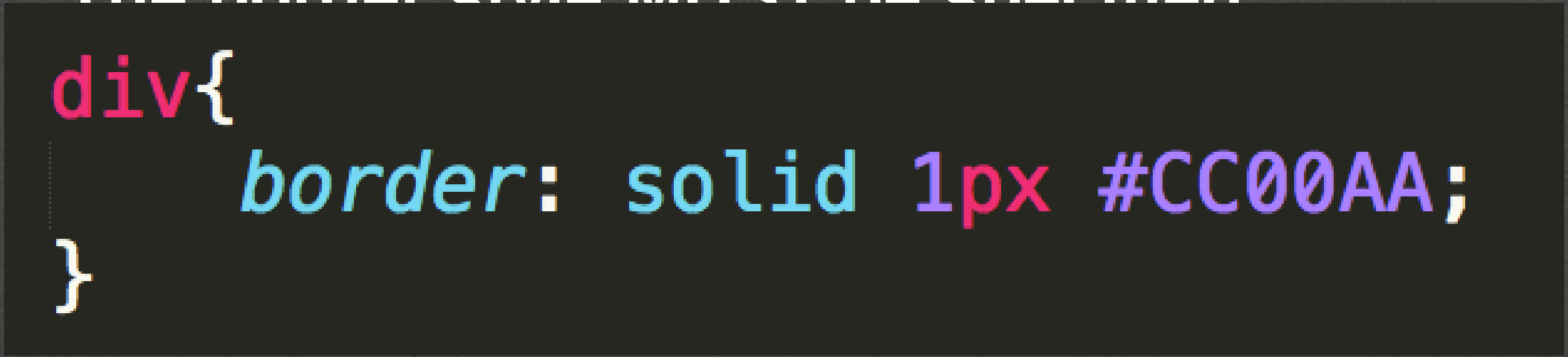
Sizing your elements

Height and Width

- The default width of inline elements is the content
- Elements that are not inline can take width and height properties – we saw this in the Display lecture.

Border

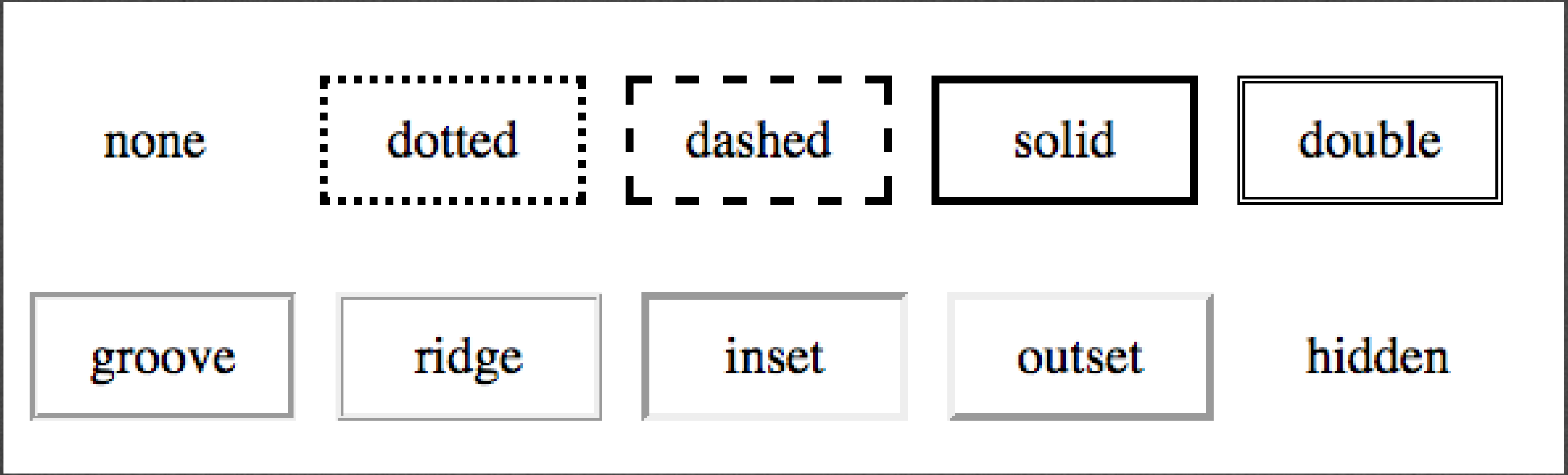
- Any element can have a border around it
- border property specifies *style*, *width*, and *color*
- The border style MUST be specified



```css
div{
	border: solid 1px #CC00AA;
}
```

Border-style

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

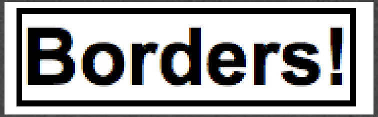


Border width and color

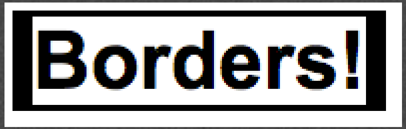
- **Width**
    - **Set in pixels or *thin*, *medium*, or *large***
- **Color**
    - **Name - “blue”**
    - **RGB – rgb(0,0,255)**
    - **hex - #0000FF**
    - **transparent**

Specifying Individual Sides

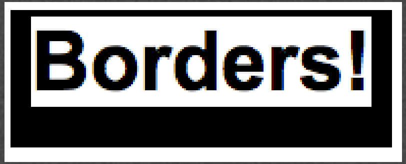
```
border-width: 3px;
```



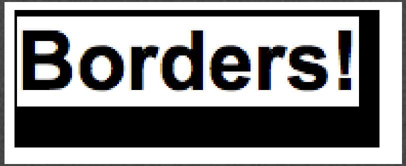
```
border-width: 3px 10px;
```



```
border-width: 3px 10px 20px;
```



```
border-width: 3px 10px 20px 1px;
```



Margin

- **Margin is additional space <u>*outside*</u> your border – between you and neighbor**
- **Positive margin**
  - **element moves right/down**
- **Negative margin**
  - **element moves left/upward**

Padding

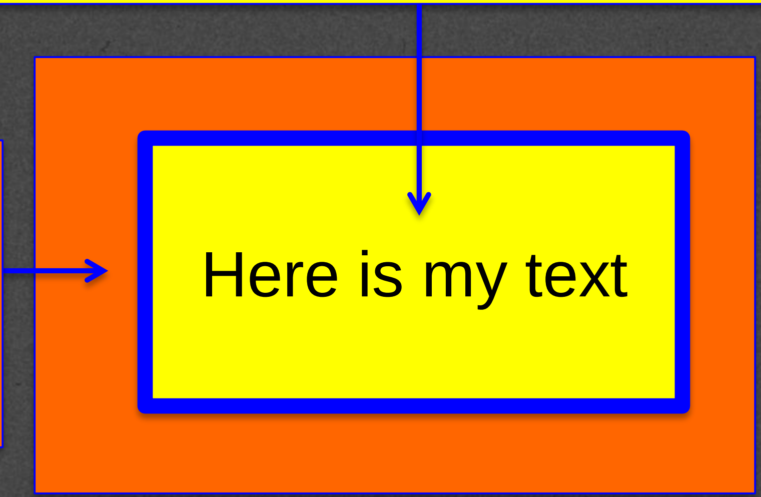
- **Padding is additional space *between* the element and its border.**
- **Positive padding**
  - **border moves outward from element**
- **Negative padding**
  - **border moves over the element**

Margin and Padding

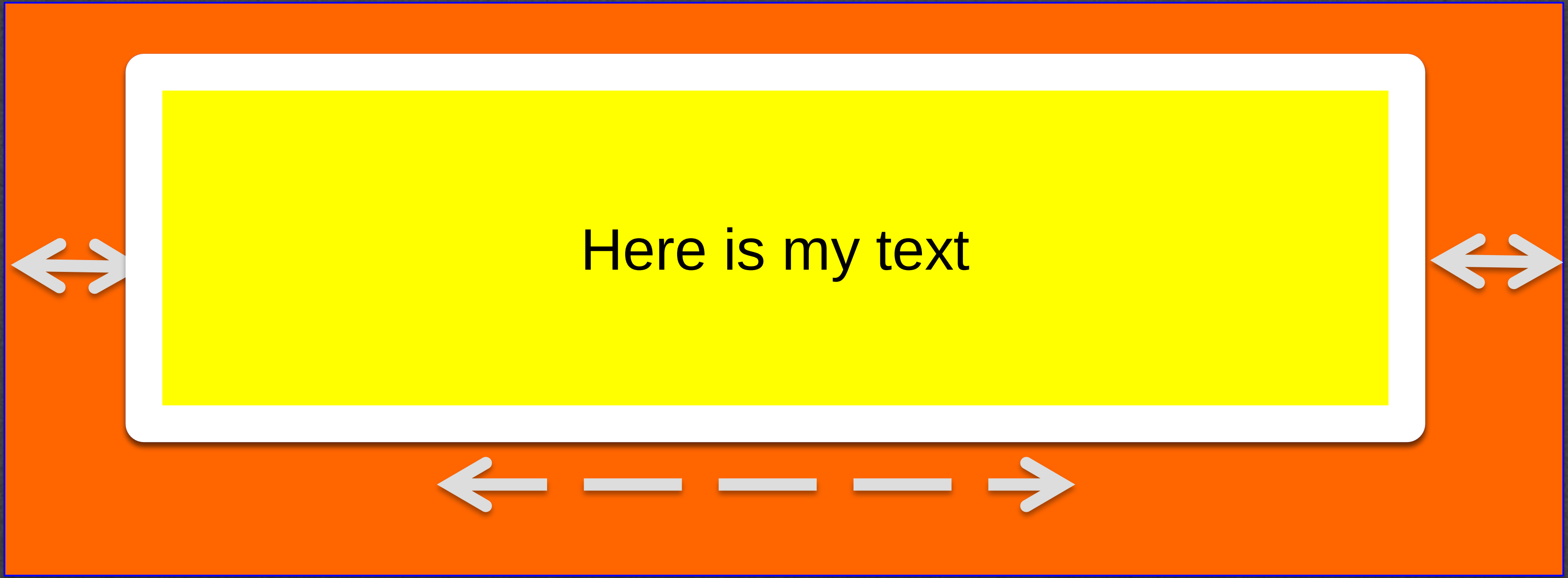
- Neither takes a color (transparent)
- Can also be defined in 1 - 4 values like border

padding
The empty space between the start of the element and the start of the text

margin
The space between the edge of the screen and the element



Additive Height and Width



margin + border + padding + width = actual width

What is the width and height?

```
div{
        width: 100px;
        height: 50px;
        padding: 10px;
        margin: 5px;
        border: 1px solid black;
}
```

width = 132px

height = 82

Centering an Element

- To horizontally center an element use:
  - margin: 0 auto;
- But…
  - The element must display: block
  - The element must not float
  - *The element must not have a fixed or absolute position*
  - The element must have a width that is not auto

box-sizing

- box-sizing takes some of the “math” out
- Options:
  - content-box: default additive
  - border-box: width takes content, padding, and border into consideration

Measurements

- Absolute – set to a specific size
  - px, mm, cm, pt, ….
- Fluid – sets size relative to surrounding elements
  - %, vw, vh
  - em (for font): 1em is current size, .75 is 75% of the current size
  - rem (for font): 1rem is current size of root element

Example

Review

- Design sketches should be done with box model (margin, border, padding, content) in mind.
- Use box-model to reduce complexity
- Margin must always be considered
- Use fluid sizes for best viewing

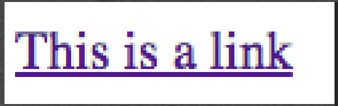
Acknowledgements/Contributions

BY NC

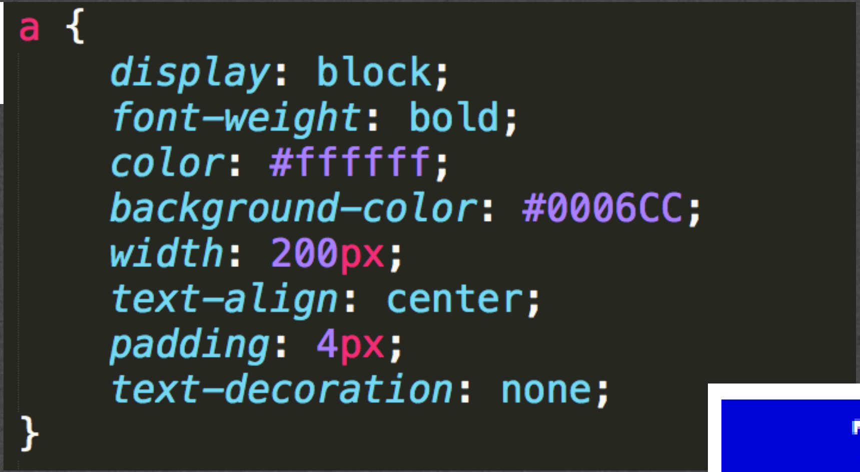
Styling Links and Lists

Anchor Links

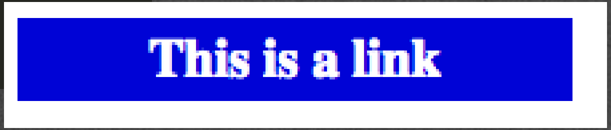
- Links can take on all of the usual styles as well as *text-decoration*



This is a link



```
a {
    display: block;
    font-weight: bold;
    color: #ffffff;
    background-color: #0006CC;
    width: 200px;
    text-align: center;
    padding: 4px;
    text-decoration: none;
}
```



This is a link

“Buttons”

- Many designers try to make their links look like buttons.
- Be semantic, if you want a button use the <button> element instead.

```
<button>Click Me!</button>
```

States

- Some links are blue, some are purple, etc. Why???
  - a:link: a normal, unvisited link
  - a:visited has been visited
  - a:hover activated by mouse (touchscreens….?)
  - a:focus activated with the keyboard
  - a:active is being clicked

Precedence of Rules

- a:hover MUST come after a:link
- a:visited and a:active MUST come after a:hover

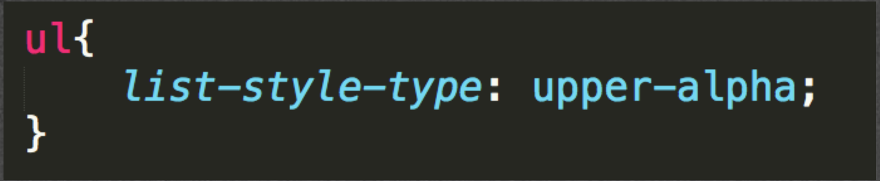
Example

Styling Lists

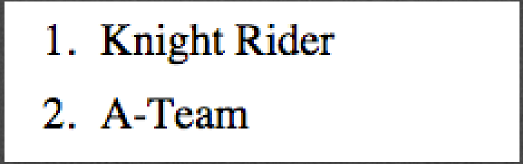
- Number of properties beyond font, margin, etc.
  - list-style-type
  - list-style-image
  - list-style-position
  - list-style

list-style-type

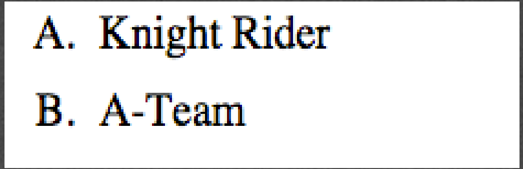
- list-style-type
  - ordered lists



```
ul{
    list-style-type: upper-alpha;
}
```



1. Knight Rider
2. A-Team



A. Knight Rider
B. A-Team

- *lower-roman, upper-roman, decimal, decimal-leading-zero, upper-alpha, lower-alpha, hebrew, armenian, …..*

List styles

- list-style-type
  - unordered lists
    - *Override the default marker with circles, discs, or squares*
- list-style-image
  - Use a custom image instead of traditional marker

```css
ul {
    list-style-image: url('icon.gif');
    }
```

Example

Review

- At this point you have learned how to write rules for the *tags*.
- Embrace the many tools that are available to help you design your site.
- http://chrispederick.com/work/web-developer/
- http://css3generator.com/
- Do web search for “Developer Tools”

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

Advanced Selectors

Styling Specific Objects

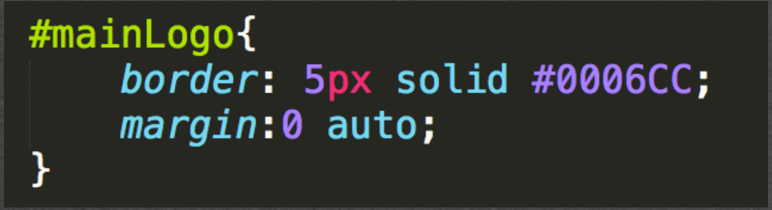
- We have focused on *type* selectors.
- What if you don’t want to style *all* of the links, just some? Or just some of the lists?
- CSS gives you options

CSS Selectors

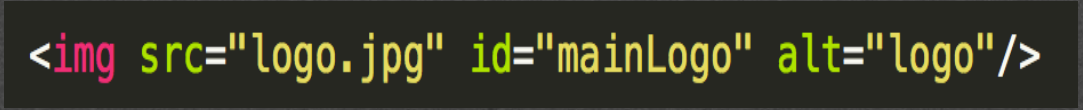
- Some selectors follow the DOM
- Descendant selectors (nav a)
  - Style all of the anchor links inside a nav tag
- Child selectors (nav > a)
  - more constraining The anchor elements must be a child of the nav, no intermediate tags, e.g. paragraph
- Adjacent sibling (h1 + o)
  - elements must be at same level and follow each other

id Selectors

- # id selector
  - Used to identify a single element in the DOM.
  - Was used extensively for <div id = “header”>, <div id=“footer”>, etc.
  - There is a small movement to move the use of id OUT of CSS



```
#mainLogo{
    border: 5px solid #0006CC;
    margin:0 auto;
}
```



```
<img src="logo.jpg" id="mainLogo" alt="logo"/>
```

class Selector

- . class selector
  - Used to identify an element in the DOM that is part of a special class of items
  - Think of thumbnail images, all of the links that are in the navigation, your social media images,

```
.thumb{
    border: 1px solid #0006CC;
    width:20%;
}
```

```
<img src="cat.jpg" class="thumb" alt="Joe"/>
<img src="dog.jpg" class="thumb" alt="Bacon"/>
<img src="bird.jpg" class="thumb" alt="Tweety"/>
```

classes vs. ids

- Syntax is "." and "#"
- classes can be used multiple times
- id should be unique
- Think of images and navigation bars
  - Format numerous (but not all) images the same way
  - Visually signify the current page

Example

Narrowing the Scope

- As you get more advanced pages, you will want to narrow the scope of the of action
- p.main → paragraphs using main class
- header img.special → paragraphs inside header that use special class

Expanding the scope

- You can combine elements with a comma
  - p, h1, #main, .special{…rules to apply to all of them…}
- Review : What happens when there are multiple rules for the same selector?
  - When there are conflicts, use the one processed most recently
  - UNLESS a rule has !important

More Attribute Selectors

- Universal
  - * applies styling to every element on the page
  - Ackk!! Try this!
- Attribute Selectors
  - a[href=‘info.html’]
- PseudoClasses
- Pseudo Elements

Attribute selectors

- You may want to search the DOM for certain elements that have an attribute you are looking for
  - All the images that use gif files…..
  - All of the images that have empty alt text….
  - All of the links that go to government sites….

Using Operators

- **Operators can be used to find those attribute values you are looking for**
  - ^ : *match the beginning exactly*
    - *a [href^='http://umich']*
  - $: *match the end exactly*
    - *img[src$ = '.png'] → apply to .png images*
  - * : *wildcard*
    - *a [href*='umich']*

Example

Whew!!!

- We have actually covered a lot in this short video
- Know that each of these ideas can merge. One element can have many classes and ids associated with it

```
<li class="special early dark" id ="main"/>
```

- Browser "starts at the top" and applies each rule, sometimes overriding earlier rules.

The Good News

- You can use style sheets from others to style your code, just by adding class!!
- You can override style sheets from others just by rewriting the class, or making your own version of it and linking it last.

Review

- Type selectors can be combined to narrow the scope of where rules are applied
- An id is used to specify a specific element in a page
- Classes can be used to associate elements that should treated in a similar manner

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

Browser Capabilities

Designing for consistent appearance

Browsers Differ

- Even though browsers are moving to a consistent implementation of HTML, they differ in display and adherence.
- It is your responsibility to make sure your page works for a wide audience.

Handling Stylistic Differences

- "Easiest" way to eliminate browser differences is to use a default style sheet
- Default style sheets reset all of the values for the page
- Will make your page look worse!

Handling Unsupported Properties

- Not all browsers support all HTML5 tags
- Not all browsers support all CSS3 properties
- Browser prefixes (or vendor prefixes) provide a quick fix for handling unsupported CSS3 options.

Browser Prefixes

- -webkit-: Android, Chrome, iOS, Safari
- -moz-: Firefox
- -ms-: Internet Explorer
- -o-: Opera

Often Unsupported Properties

- column-count
- border-radius
- gradient
- Sites such as http://caniuse.com/ will tell you when you need to use prefixes

Example

Automated Ways to include Prefixes

- For now, add the prefixes by hand
- There are ways to automate the addition of prefixes
  - Editor add-ons (You have most of the control)
  - Use outside programs to dynamically add appropriate prefix based on browser

Review

- Default style sheets remove stylistic differences
  - Should default style sheet be internal or external?
  - Where should it go in relation to other style sheets?
- Browser prefixes can help remove some differences caused by unsupported options
  - Shouldn’t be overused

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

Designing for Accessibility

POUR

Overview

- The content of your page should be in the HTML.
- It is tempting to add content via colors, images, etc.
- Follow the POUR guidelines
  - Perceivable, Operable, Understandable, Robust

Perceivable

- Provide text alternatives for images
- Provide captions and transcripts for video and audio
- Use correct semantic markup so content can be presented in different ways
- Make it easier for users to see content by using good color contrast

Operable

- *All functionality available from the keyboard!*
- Users have control over timing and limits
- Do not cause seizures (don’t flash content)
- Provide ways to help users navigate, find content, and determine where they are

01.01 Welcome

Understandable

- Economical and plain use of language
- Text supplemented with illustrations, videos, and other formats where appropriate (i.e., use good Universal Design)
- Navigation, information structure are discernable and consistent
- Make pages operate in predictable ways
- Help users avoid and correct mistakes

Robust

- Is your site functional across various technologies (smart phone, screen reader, laptop, pensticks, etc..)?
- Syntax errors that don’t affect visual presentation may hamper assistive technology and accessibility tools
- Adhering to W3C standards ensures future compatibility
- Validate your code at validator.w3c.org and

Review

- Accessibility starts with proper HTML tags
- Styling can actually make it HARDER for some people to access the information
- Get into the early habit of utilizing accessibility tools
- “Cool” new style should not be at the cost of accessibility

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

Homework Two

Adding more style with an external CSS

Objective

- Create your own unique style sheet that will be used by three different html files.

Getting Started

- You must complete the first Peer Graded Assignment to begin this one.
- You can alter your previous styling choices but I assume those changes are complete

Before

http://www.intro-webdesign.com/CSS/assignment-2/index.html

http://www.intro-webdesign.com/CSS/assignment-2/teams.html

http://www.intro-webdesign.com/CSS/assignment-2/history.html

After

http://www.intro-webdesign.com/CSS/assignment-2/index.jpg

http://www.intro-webdesign.com/CSS/assignment-2/teams.jpg

http://www.intro-webdesign.com/CSS/assignment-2/history.jpg

You must leave the HTML alone

Create a single style sheet that updates:

- body, header, h1, the links, the active, left, and right classes, and the images in the left class.

body

- Change the padding and margin
- Feel free to adjust the size of the text, but it is not required

header

- Change background color
- Add background image, using the image provided in the css folder
    - If you have trouble with this part, keep an eye out for sample code in cal-table.css. The key is to master the folder structure

h1

- Change the font color
- Increase the font size

The navigation links

- Style ONLY those links in the nav element.
- You don’t need to recreate my example exactly, but they should be spaced apart.
- Make sure to round the borders and removed the underline.

active class

- Use the background color and font color to create an active class.
- This class is used to distinguish the current page from the others

left and right class

- These two elements should be side-by-side.
- Change the background color for the left class.

Images

- Put the images back in and put a border around them
- Make sure that they are centered within the .left class
- Put spacing between them on the top and bottom too

Peer grading

- Grades will be based on level of completion
- Some aesthetics will come into play this time. It is important that things are not “squished” together
- Proper standards do apply
- You can specify your preferred screen size for grading.

Acknowledgements/Contributions

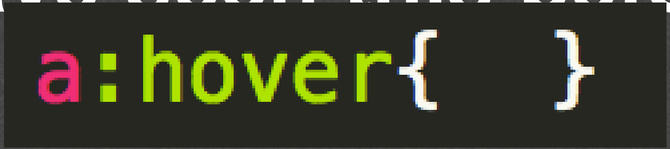
Initial Development: Colleen van Lent , University of Michigan School of Information

Pseudo Classes and Elements

Designing for consistent appearance

Pseudo-Classes

- Elements that are dynamically populated or dependent on tree structure
- You have seen this before…



```
a:hover{  }
```

Types of Pseudo-Classes

- Link
  - :link, :visited
- User Action
  - :hover, :active, :focus
- Forms (interfaces)
  - :enabled, :checked, :disabled

Types of Pseudo-Classes

- **Structural/Positional**
  - **:first-child, :last-child, :nth-child(), :only-child**
  - **:first-of-type, :last-of-type, :only-of-type**

```
li:first-child{   }
li:nth-child(4){   }
p:empty{  }
img:only-of-type{   }
p:last-of-type{   }
```

Example

Pseudo-Elements

- These elements aren’t part of the DOM
- Can be used to style specific (unique) parts of the page

Types of Pseudo-Elements

- Textual
    - :first-letter, :first-line
- Positional/Generated
    - :before, :after
- Fragments
    - ::selection

Example

Review

- Pseudo-elements and classes are just one more way to add style to your page
- I haven’t covered every combination so make sure to do some investigation on your own

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

Transitions

Transitions

- When elements transition from one state to another, you can alter their appearance
  - If you hover over the link, change the color
  - If an imag nes into focus, change the size,

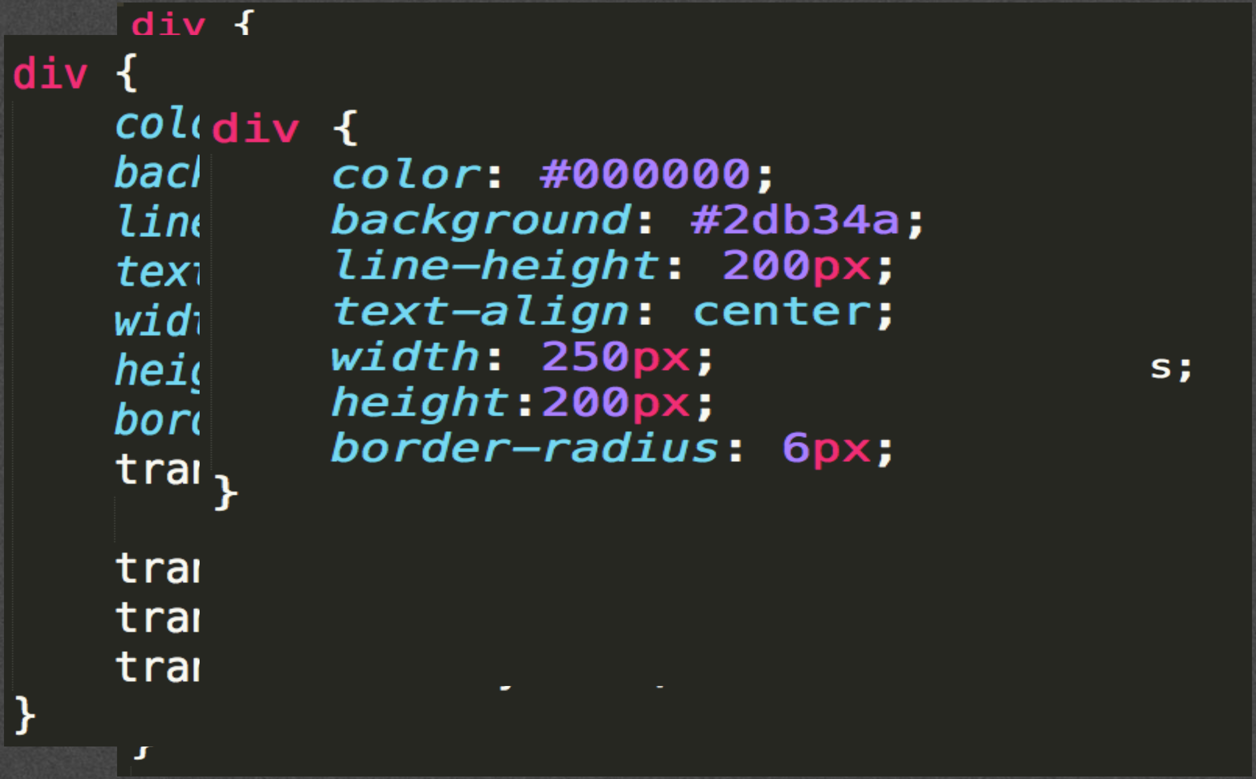
The Properties

- transition-property
  - What is it you want to change? (size, color, position, etc.)
- transition-duration
  - How long should each transition last?
- transition-timing
  - Should it be a smooth transition (linear)? Or different?
- transition-delay
  - How long should the wait be before the transition begins?

Setting up

1. Define your element
2. Choose the elements for transition
3. Define the new values
   - You must combine this step with a pseudo-class

Example (CSS3-transitions)



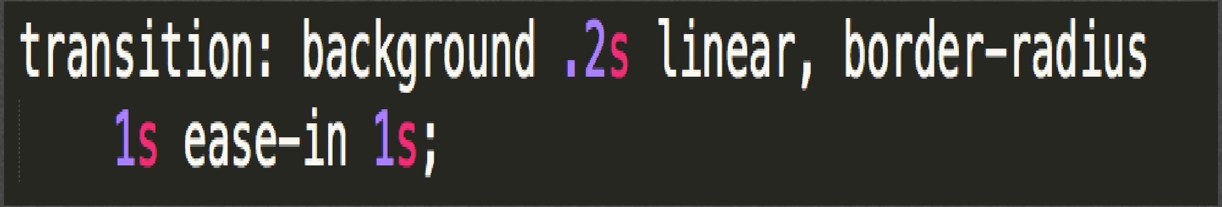
```
div {
    col
    bacl
    line
    text
    widt
    heig
    bord
    tra

    tra
    tra
    tra
}
```

```
div {
    color: #000000;
    background: #2db34a;
    line-height: 200px;
    text-align: center;
    width: 250px;
    height:200px;
    border-radius: 6px;
}
```

Using Shorthands

- If you have multiple properties transitioning, you can use shorthand:



```
transition: background .2s linear, border-radius
    1s ease-in 1s;
```

Review

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

Transforms

Transforms

- Provide option for changing the appearance of elements
- Two-dimensional
- Three-dimensional

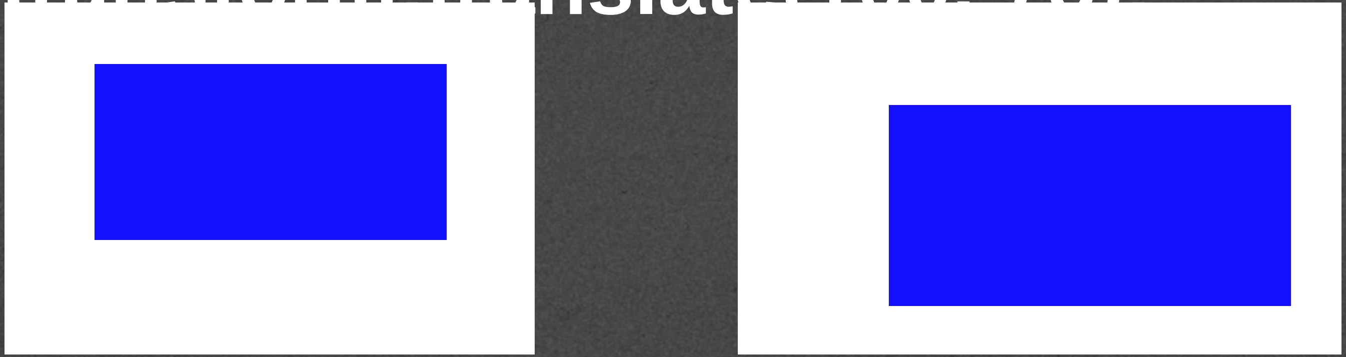
2D Transform Options

- Options
  - translate
  - rotate
  - scale
  - skew
  - matrix

translate

- transform:translate(x, y);
  - move x pixels to the left/right and y pixel up/down

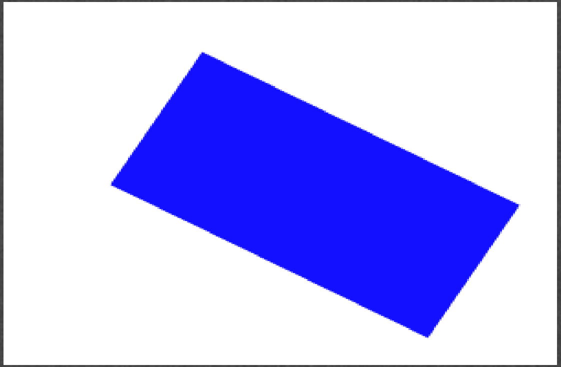
transform:translate(100, 75);



rotate

- transform:rotate(deg);
    - Rotate/”spin” the element a certain number of degrees

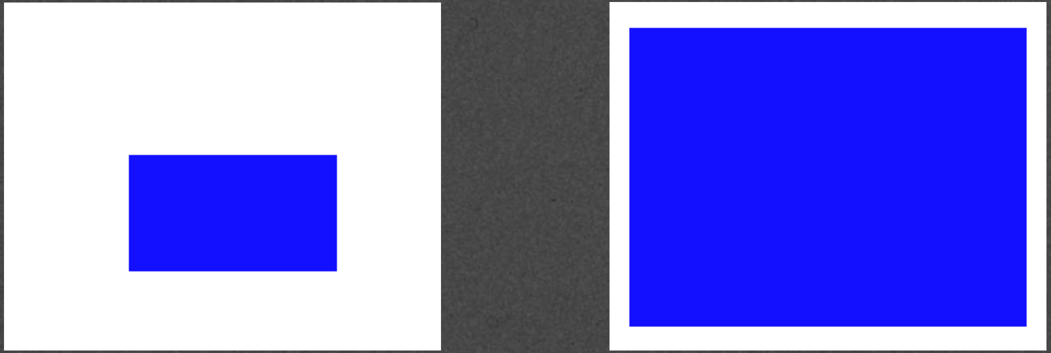
transform:rotate(30deg);



scale

- **transform:scale(width, height);**
  - **Change the width and height of the element**

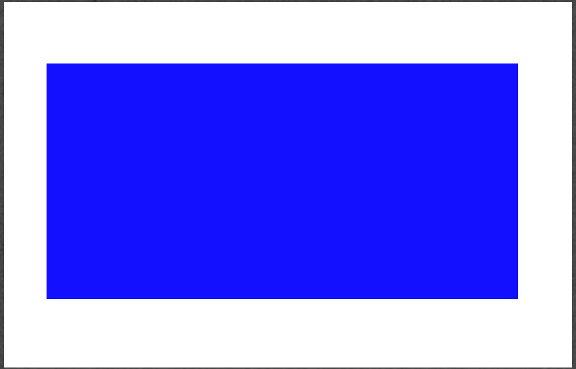
transform:scale(2,3);

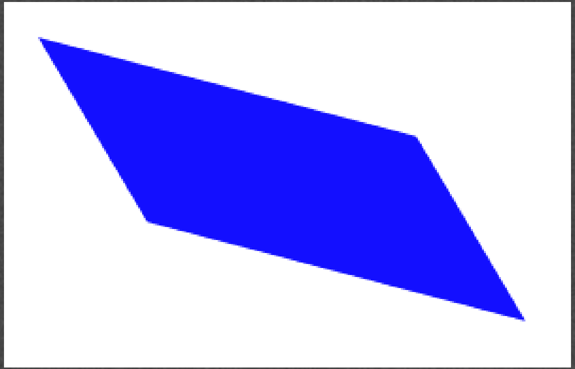


skew

- transform:skew(x-angle, y-angle);
  - Rotate the element a certain number of degrees along the x and y axis

transform:skew(30deg, 15deg)





matrix

- matrix() - combines all of the 2D transform methods into one

3D rotate

- You can rotate along the x, y, or z dimension along a given degree
- transform:rotateY(deg)
- transform:rotateX(deg)
- transform:rotateZ(deg)
- transform:rotate3d(x, y, z)

Others

- 3D scale
- 3D translate

Review

- Transforms are one more way to modify the look of your page.
- Often combined with state changes
- Will typically require browser prefixes.

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

Positioning

Positioning!

- Putting elements where you want them can be time-consuming and frustrating
- Why not tables?

Position Properties

- The four position properties are:
  - static
  - relative
  - absolute
  - fixed

- Position properties are modified by the properties: top, right, bottom, left

Static

- Default value for elements
- Place in the next available position
- Not affected by the top, bottom, left, and right properties.

Relative

- Positioned “relative to itself”
- Take the static position, but add offsets.
- The new positioning does NOT affect any other element. It is possible to move an element and leave a big hole where it would have been.
- Relatively positioned elements are often used as container blocks for absolutely positioned elements.

Absolute

- Element is removed from the document flow and positioned relative to it's *nearest ancestor* (or the root)
- Other elements behave as if element does not exist
- Can end up on top of another element

Fixed Position

- Positioned relative to the *browser window*
- Will not move, even if the window is scrolled
  - IE7 and IE8 support the fixed value only if a !DOCTYPE is specified
- Think of popup boxes that wont’ go away!!!
- Or a navigation bar that is always visible on the top

Example

Z-index

- Multiple elements may be placed in the same position.
- z-index is a numeric value, positive or negative that dictates stacking order

Example

Review

- Positioning elements is key to achieving desired layouts
- Proper planning will make this easier

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

Accessible Navigation

Navigation

- Navigation is a critical aspect of accessibility
- Sighted users have tried and true visual cues to orient them on a page
  - Banner
  - Search box
  - Main navigation box
  - Content well
- Blind and low-vision users rely on proper coding of page for orientation

What if you can’t see?

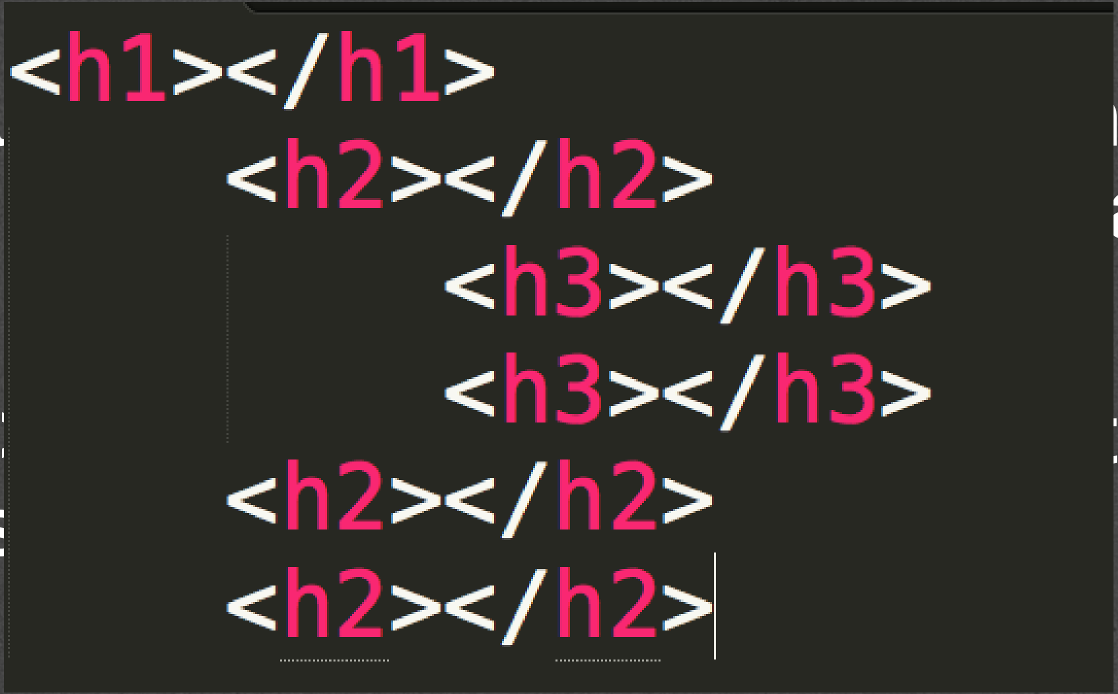
- Title of page lets you know what page you’re on when page loads
- Proper heading placement and hierarchy conveys organization of page and allows SR users to skip navigation
- Link descriptions convey content of page and organization of site

Proper <h1> heading

- Screen readers can find and list headings
- <h1> heading uniquely identifies the page in the website
- Should be placed directly in front of the main content of the page
- The <h1> header should also match at least a subset of the the page <title>

Proper heading hierarchy

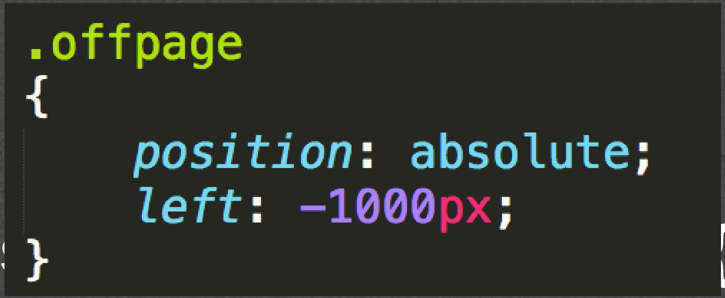
- Hea   ested
  to c   age
- <h2   the <h3>
  tags



```
<h1></h1>
    <h2></h2>
        <h3></h3>
        <h3></h3>
    <h2></h2>
    <h2></h2>
```

Off-page headings

- Useful when you want to give SR users a navigational aid without cluttering presentation
- Use CSS to position headings off-page



```
.offpage
{
    position: absolute;
    left: -1000px;
}
```

- Don’t us {visibility: hidden}

Meaningful link text

- Screen readers can find and list links
- Descriptions for the links must be meaningful out of context, via tabbing or presented in a list
- Don’t use “here”, “click here”, “read this”, and “more”
- Don’t use URL as a link description—will sound like gibberish, unless very short and intuitive

Review

- How easy is it to navigate your page?
- What would happen if the colors weren’t there?
- What would happen if you couldn’t use a mouse?
- Plan for everyone

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

Final Project

Objective

- Create your own unique style sheet that will be used by three different html files.

Getting Started

- You must complete the second Peer Graded Assignment to begin this one.
- You can alter your previous styling choices but I assume those changes are complete

Before

http://www.intro-webdesign.com/CSS/final/index.html

http://www.intro-webdesign.com/CSS/final/teams.html

http://www.intro-webdesign.com/CSS/final/history.html

After

http://www.intro-webdesign.com/CSS/final/index.jpg

http://www.intro-webdesign.com/CSS/final/teams.jpg

http://www.intro-webdesign.com/CSS/final/history.jpg

You must leave the HTML alone

- Create a single style sheet that styles the table
- You will use pseudo-elements and pseudo-classes

table

- Style at a minimum:
  - Font – multiple families, size, weight, and line-height
  - Width – don’t take up the entire space
  - Margin – center the table

table heading

- Change background color
  - Try to use a gradient
- Add a border
  - Round only the top two corners.
  - Make the bottom border thicker
- Add at least one other property

table rows

- Set the opacity to a value of about .8
- Left align the first column, center the other two
- Set the opacity to a value of 1 when the element hovered upon.

td

- Change the padding, font color, background color, and set the border radius to 2px.
- Optional: use text-shadow

Peer grading

- Grades will be based on level of completion
- Proper standards do apply
- You can specify your preferred screen size for grading.

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information

Closing

Congratulations

- You have come a long way from the plain pages we began with!

What next?

- Consider creating your own site
- Continue to practice your skills
    - You are ready to join a Meet Up, or offer your skills as a TA at workshops.
- Begin to learn about using JavaScript to add interactive elements to your site
- Learn more about Responsive Design

Thank you!

Acknowledgements/Contributions

Initial Development: Colleen van Lent , University of Michigan School of Information