CSS3

01.02 - Adding Style to your Pages!

Adding style

(1) **Style attribute** → Violated separation of content/style

CSS defined generic rules that can apply to multiple elements

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

(2) Internal Style Sheet

Styling is defined within <head>

Rules are defined within <style>

(3) External Style Sheet

Put rules in an external file (don’t use the style tag

A link to the style sheet is put in the head section.

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 one selector has more than one rule in the same file?
→ The **most recent rule** has precedence

!important

It is possible to override later rules, use !important

01.04 - Styling text

Custom fonts

To expand beyond "web-safe" fonts use *@fontface*

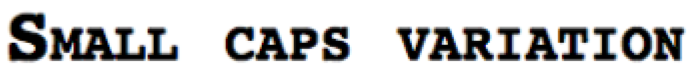
```
@font-face {
  font-family: myFont;
  src: url('Colleen.ttf');
}

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

font-variant

normal

small-caps



Small caps variation

line height

Adjusts the space between the lines of text

01.04 - Styling text

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!

`Inline-block` : same as inline, but accepts height and width

`none`: removed from page; Still in DOM, but not visual (even to SRs), not take space

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!

Overflow

visible	Can cause text to show up “on top” of other text
hidden	Hides anything that goes beyond bounding box
scroll	Gives horizontal and vertical scrollbars
auto	Adds scrollbars as needed

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

`visibility: hidden` - still part of DOM, take up place

`display: none` - remove from DOM, not take space

02.01 - Box Model

Height and Width

The default width of inline elements is the content

Elements that are not inline can take width and height properties,
use display: inline-block to apply width and height.

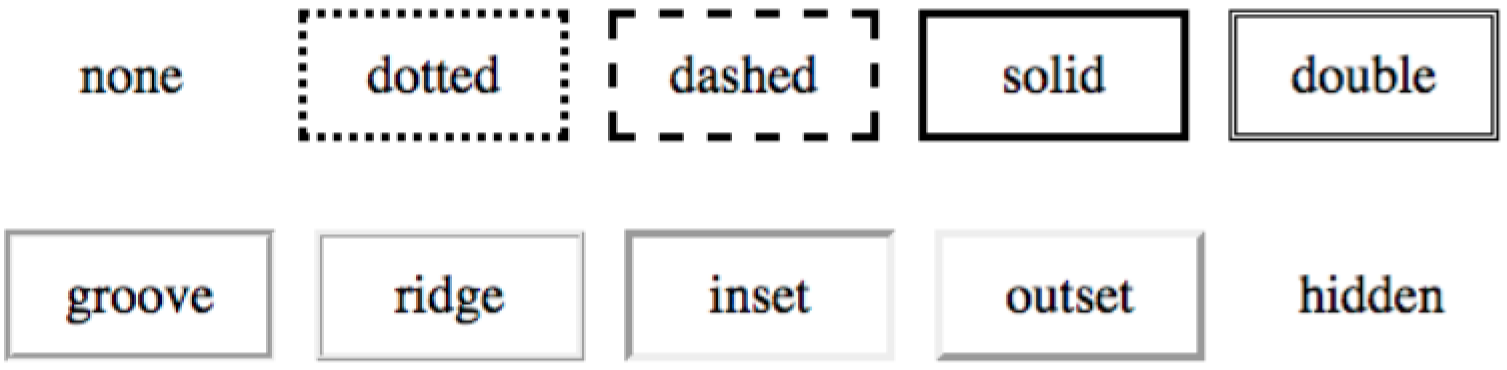
Border

Any element can have a border around it

border property specifies style, width, and color

The border style MUST be specified

Border style



Margin

Margin is additional space outside 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

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

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

02.03 - Styling Links and Lists

Links

Anchor links

Buttons: Many designers try to make their links look like buttons.

States:

– 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`

Lists

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

list-style-type:
+ ordered list: upper-alpha, lower-alpha, lower-roman, upper-roman, decimal, decimal-leading-zero, hebrew, armenien,...
+ unordered list: override the default marker with circles, discs, square,...

list-style-image: url('icon.gif')

02.04 - Advanced selectors

CSS Selectors

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

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

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, etc.…

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…
}

More Attribute Selectors

1. **Universal**: * applies styling to every element on the page
2. **Attribute Selectors:** a[href=‘info.html’]
   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’]
3. PseudoClasses
4. Pseudo Elements

02.05 - Browser Capabilities

Handling Unsupported Properties

→ `Browser prefixes` (or `vendor prefixes`) provide a quick fix for handling unsupported CSS3 options.

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

Often Unsupported Properties

• column-count
• border-radius
• gradient

02.06 - Designing for Accessibility

POUR guidelines:
Perceivable, Operable, Understandable, Robust

1. **Perceviable**

- 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

2. **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

3. **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

4. **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

03.01 - Pseudo Classes and Elements

Pseudo-Classes

Elements that are dynamically populated or dependent on tree structure

Types of Pseudo-Classes

- Link: :link, :visited
- User Action: :hover, :active, :focus
- Forms (interfaces): :enabled, :checked, :disabled
- Structural/Positional:
  + :first-child, :last-child, :nth-child(), :only-child
  + :first-of-type, :last-of-type, :only-of-type

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

03.02 - Transitions

The Properties

- transition-property
- transition-duration
- transition-timing
- transition-delay

Using Shorthands

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

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

03.03 - Transforms

2D Transform Options

- translate
- rotate
- scale
- skew(x-angle, y-angle) → :skew(30deg, 15deg)
- matrix

3D Transform Options

3D rotateL 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)

3D scale, 3D translate

03.04 - Positioning

The 4 position properties are

1. static

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

2. 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.

3. 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

4. **fixed**

• Positioned relative to the browser window
• Will not move, even if the window is scrolled

Z-index

04.01 - Accessible 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