

---

CSC309H1S

# Programming on the Web

Winter 2023

## Lecture 3: Cascading Style Sheet

Instructor: Kuei (Jack) Sun

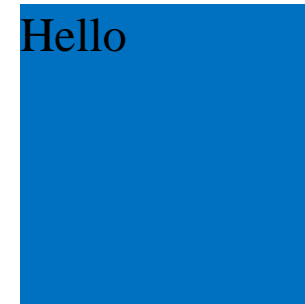
Department of Computer Science  
University of Toronto

# Web Standards Model

- Separation of content (HTML) and appearance (CSS)
  - Modern websites are dynamic – content is frequently updated
  - Most websites strive for consistency in appearance
    - Improves look-and-feel and brand recognition
  - This allows content and appearance to be updated independent of each other
- Other benefits
  - Accessibility
    - Simplifies the work of screen reader
  - Device compatibility
    - Web page can be rendered nicely on difference devices
  - Search engine
    - Helps search engine with parsing your web pages; avoids misclassification

# Cascading Style Sheet

- Describes the presentation of a document written in markup languages
- CSS **property**
  - Defines the style or behaviour of an element
  - Full list of properties: <https://www.w3schools.com/cssref/index.php>
  - Syntax
    - *property name : property value(s) ;*
- Where to specify properties?
  1. Inline style
    - An attribute named “style”
    - Style only applies to this element

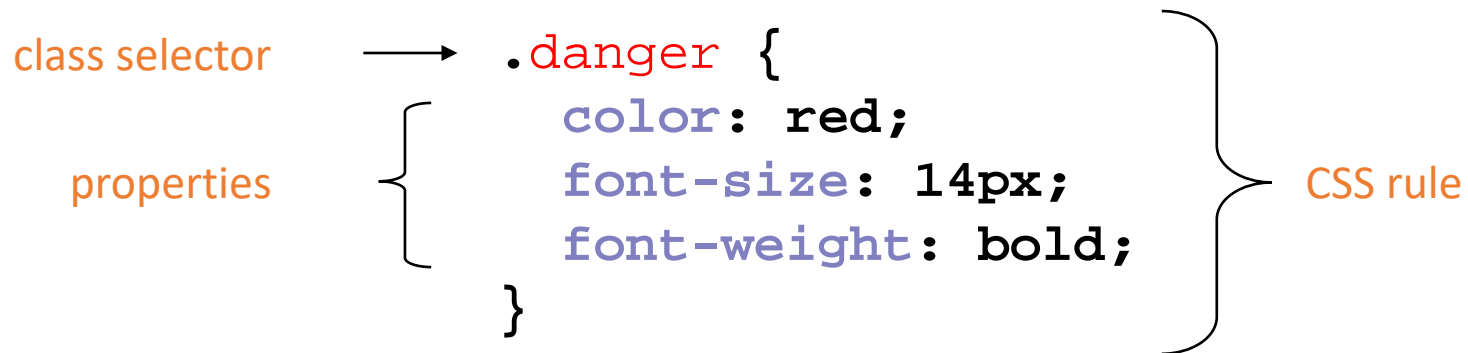


```
<div style="width:200px;height:200px;background-color:blue;">Hello</div>
```

# Where to specify properties?

## 2. CSS rule

- One or more properties that apply to a one or more elements
- CSS selector
  - Determines which elements are targeted
- Syntax: *selector { properties ... }*



- Can write CSS rules inside `<style>` element, or in a `.css` file

# Where to write CSS rules?

- `<style>` tag
  - Needs to be inside the `<head>` element
  - Not recommended except during development or debugging
- css file
  - Needs to be “imported” in the html file
  - Can import multiple css files; provides basic modularity
  - `<link>` tag should go in the `<head>` element
    - `<link rel="stylesheet" href="style.css">`
- What does “cascading” mean?
  - Multiples rules can affect the same element.

# CSS Cascade

- Given the following paragraph element

```
<p class="danger">This is an important message.</p>
```

- and the following CSS rules

```
.danger {  
  color: red;  
  font-size: 20px;  
  font-weight: bold;  
}
```

```
element selector  
↓  
p {  
  font-family: san-serif;  
  font-size: 1rem;  
  background-color: aliceblue;  
}
```

↑  
some colors have names

- Output:

**This is an important message.**

- What happens if two rules override the same property?

# CSS Specificity

- A less specific rule is overridden by a more specific rule
  1. Order of appearance (later is better)
  2. Elements and pseudo-elements
    - Example: `p`, `h1`, `::before`
  3. Classes, pseudo-classes, and attribute selectors
    - Example: `.danger`, `:hover`, `[name]`
  4. IDs
    - Example: `#footer`
  5. Inline style
  6. `!important` rule
    - Example: `font-size: 1rem !important;`
- See <https://wattenberger.com/blog/css-cascade> for more detail

# CSS Selector

- Element selector

- Syntax: `tagname`

- Example:

```
p {  
  color: blue;  
}
```

- Class selector

- Syntax: `.classname`

- Example:

```
.danger {  
  color: red;  
}
```

- ID selector

- Syntax: `#idname`

- Example:

```
#title {  
  color: black;  
}
```

- Pseudo-class selector

- Syntax: `:pseudoclass`

- Selects element in a special state

- Example

```
:hover {  
  color: green;  
}
```



# Advanced CSS Selector

- See wc3schools for a list of all CSS selectors
  - [https://www.w3schools.com/cssref/css\\_selectors.php](https://www.w3schools.com/cssref/css_selectors.php)
- Attribute selector
  - Syntax: `[attrname]`
    - Select element with attribute name
  - Example:
    - `[href] { color: rebeccapurple; }`
- Pseudo-element selector
  - Syntax: `::pseudoelement`
  - Example:
    - `::first-letter { font-size: 200%; }`

# Combining CSS Selectors

- AND condition

- Syntax: join multiple selectors without space in between

`p.danger`

`a:hover`

`p.danger.big`

- OR condition

- Syntax: join multiple selectors with comma in between

`h1, h2, p { ... }`

`h1, .center { ... }`

- Descendant condition

- Subsequent selector must be child or descendant of current selector
- Syntax: join multiple selectors with space in between

`.center p`

`header nav a`

- Other

- Immediate child condition

`form > input`

- Adjacent sibling condition

`div + p` (selects p after a div)

# Exercise 2

Do question 2 to 4.

# Font Properties

- color
  - Selects text color
- font-family
  - Select one or more fonts, in that order (in case former one is unavailable)
  - Web safe fonts
    - [https://www.w3schools.com/cssref/css\\_websafe\\_fonts.php](https://www.w3schools.com/cssref/css_websafe_fonts.php)
- font-style:
  - Either `normal` or `italic`
- font-weight:
  - `normal` or `bold` are most used options, others are possible

# More Font Properties

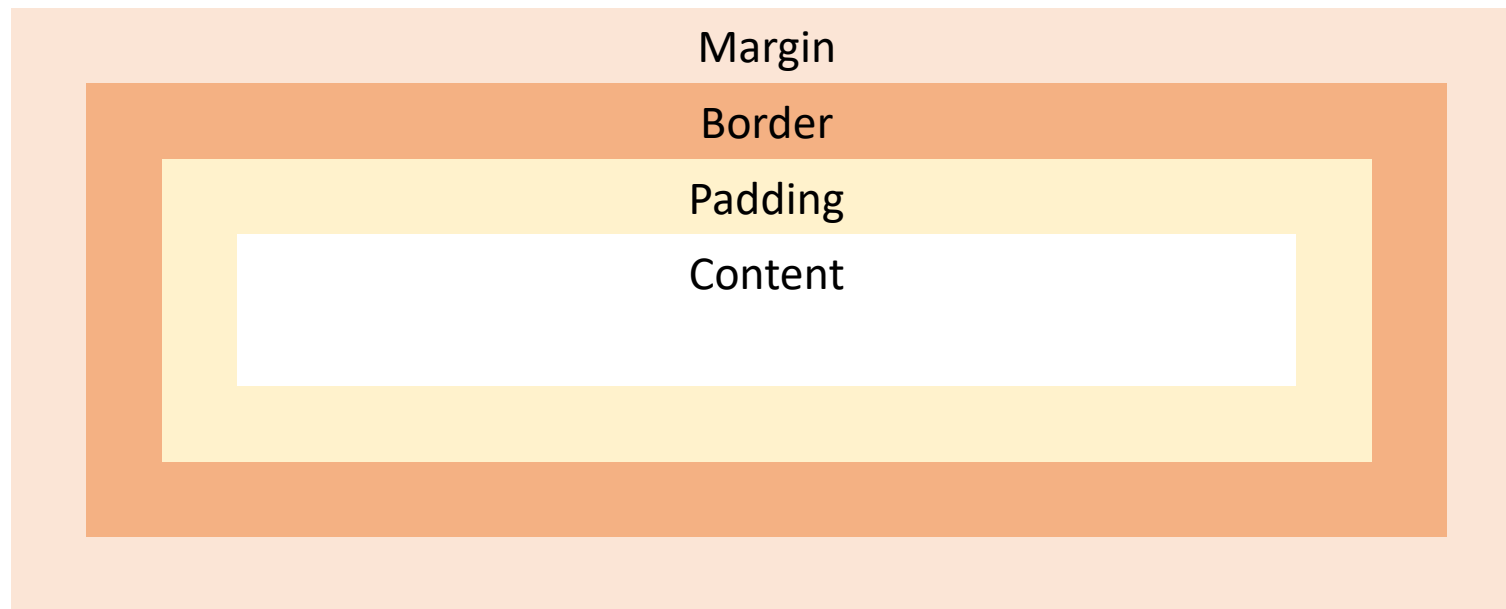
- text-decoration
  - none, underline, overline, or line-through
- text-align
  - left, right, center, justify (all lines of text are same width)
  - You likely want to use the hyphens property with justify
- text-size
  - Set size of text
- Further reading
  - font, line-height, text-shadow, text-transform, letter-spacing, word-spacing, etc.
  - <https://developer.mozilla.org/en-US/docs/Web/CSS/font>

# Units

- Used by any property that specifies size or length
- Absolute units
  - cm (centimeter), in (inch), `px` (pixels)
- Relative units
  - rem: root element's font-size (default is 16px)
  - em: parent element's font-size
  - vh, vw: current screen's (viewport) height or width
  - %: a percent relative to the size of the parent element
  - fr: fraction (of the available space)
- `calc` function: allows you to do math on different units  
`width: calc(100% - 100px);`

# Box Model

- A set of boxes that wraps around every visible HTML elements
- The width and height of an element *includes* border, padding, and content, *but not* margin



# Spacing Properties

- Many ways to specify margin/padding/border

- Using `margin`, `padding`, or `border-width`

- Specify all edges

`border-width: 1px 2px 3px 5px;` (top-right-bottom-left)

`margin: 0;` (all edges)

`padding: 1rem 2px;` (top & bottom, left & right)

- Specify specific edges (top, right, bottom, left)

- Append which edge you want to change (except border)

`border-top-width: 1rem;`

`margin-bottom: 12px;`

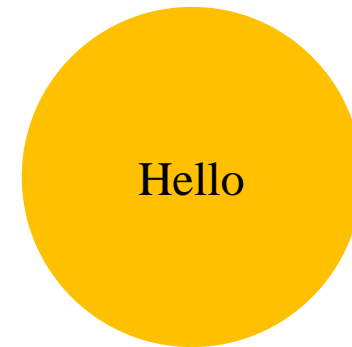
- Trick: margin can be *negative* to pull other elements closer



# Border Properties

- border-style
  - none, solid, dotted
- border-color
- border-radius
  - Adds rounded edges to element
  - Trick: can create a circle when radius is exactly half the width and height

```
p {  
  width: 400px;  
  height: 400px;  
  line-height: 400px;  
  text-align: center;  
  border-radius: 200px;  
  background-color: gold;  
}
```



# Position Property

- Specifies how to position an element
- [https://www.w3schools.com/css/css\\_positioning.asp](https://www.w3schools.com/css/css_positioning.asp)
- static
  - Default behaviour. `top`, `bottom`, `left`, and `right` properties are **ignored**.
- relative
  - Relative to its static position (where it would have been)
  - Makes it “positioned”.
- fixed
  - Relative to the *viewport*, i.e., stays in the same place on the screen.
- absolute
  - Relative to the nearest “positioned” ancestor, i.e., not static.
- sticky
  - Relative to the user’s scroll position.

# Display Property

- Specifies how to render an element and/or its children
- Some display behaviours affects how child elements are placed
- inline
  - Display the element as if it were an inline element
- block
  - Display the element as if it were a block-level element
- none:
  - Do not display this element, as if it were removed
- inline-block:
  - Same as inline, but `width` and `height` properties are allowed.

# Responsive Design

- A web design approach
  - Pages adjust themselves to “look good” on all screen sizes
- Viewport setup
  - Required to ensure viewport adjusts to current screen size

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```
- Responsive image
  - Set width property to 100%
- Responsive text size
  - Set font size to a percentage of viewport width, e.g., 10vw (10% of width)
  - Use clamp function, e.g., `font-size: clamp(1rem, 10vw, 2rem);`

# Responsive Layout

- Allows elements to be placed flexibly depending on screen size
- **float** property
  - Useful for creating magazine-like layout
  - Should *only* be used to place an element to the side of a container, while allowing other elements to flow around it
  - Should *not* be used to design page layout in modern days

Lorem ipsum dolor sit amet consectetur adipisicing elit.



Inventore, voluptatem incidunt voluptas nobis placeat facilis commodi laboriosam similique id veritatis molestias, dignissimos praesentium, autem tenetur consequatur beatae itaque. Ipsa, iure.

Morbi ornare tortor nisi, sed fermentum ipsum faucibus vitae. Aenean viverra mauris sit

# Flexbox

- Flexibly places items inside the parent element, a.k.a, a **container**
  - `display: flex;`
- Container size automatically adjusts to size of child elements
- justify-content
  - right, left (default), space-evenly, space-between, space-around
- flex-wrap
  - wrap (to the next line), nowrap (default)
- flex-direction
  - column (top down), column-reverse (bottom up), row, row-reverse

# Flexbox Properties

- align-items
  - Where to place child elements in a large container
  - center, flex-start, flex-end, etc
- align-content
  - Determines where to place each flex line (row or column) in a large container
  - center, flex-start, flex-end, stretch (default), space-between, etc.
- Trick: perfect centering

```
.container {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
}
```

Better than using vertical-align!

# Flex item properties

- Flex item
  - The direct child elements of a flex container
- flex-grow, flex-shrink
  - How much a flex item will grow or shrink. Default is 0 (no growth/shrinkage)
  - Relative to other items
- flex-basis
  - Initial length of the flex item (width if row, height if column)
- align-self
  - Overrides the container's align-items property
- Exercise 2: <https://flexboxfroggy.com/>



# Grid Layout

- Grid supports two-dimensional layout, similar to table
  - However, use grid for layout; use table for tabular data
- `.container { display: grid; }`
- grid-template-columns
  - For each column, specify a size value, e.g., 3 columns:  
`grid-template-columns: auto auto auto;`
- grid-template-areas
  - Uses named grid items to specify rows and columns  
`grid-template-areas: 'menu top top'  
                          'menu bot bot';`
- gap, row-gap, column-gap
  - Space between rows and/or columns

# Grid Item Properties

- Grid item
  - Direct child of a grid container
- grid-column-start, grid-column-end
  - Similar to `colspan` for `<td>`; allows you to span multiple grid columns
  - ```
.item1 { grid-row-start: 1; grid-row-end: 3; }
```
- grid-row-start, grid-row-end
  - Similar to `rowspan` for `<td>`; allows you to span multiple grid rows
- grid-area
  - Specify which named area this grid item belongs to
  - ```
.item1 { grid-area: menu; }
```
  - Can also be used to specify unnamed area to span both rows and columns

# Exercise 2

Do question 6.

# Media Query

- Checks the capability of the device before applying CSS rules
- Can completely change layout based on device
- Syntax: `@media type and (expressions) { CSS Rules... }`
  - *type* is one of: screen, printer, speech

- Example:




```
@media screen and (min-width: 480px) {  
    #leftsidebar {width: 200px; float: left;}  
    #main {margin-left: 216px;}  
}
```

- Mostly likely used expressions:
  - min-width, max-width

# Browser Support

- Not all browsers support the same CSS properties
- <https://caniuse.com/>
  - Tells you if a feature is supported
- Example
  - Media Queries: Range Syntax
  - Allows you to write media query like this:  
`@media (100px <= width <= 1900px)`
  - Easier to read compared to this:  
`@media (min-width: 100px) and (max-width: 1900px)`
  - Supported on Chrome 104 or higher, Firefox 63 or higher
  - Currently not supported on Safari. (as of January 2023)

# CSS Framework

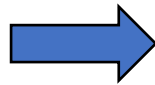
- Ready-to-use CSS libraries (may include JavaScript)
-  Bootstrap,  Tailwind CSS,  Bulma
- Provides basic and advanced interface components
  - E.g., Bootstrap modal
    - <https://getbootstrap.com/docs/4.1/components/modal/>
- Suggested for most web development project
  - Easy to use and maintain consistent style
  - Speeds up development cycle
  - Browser compatibility is (mostly) handled by the framework
  - CSS optimization is done for you

# CSS Tools

- Minifier

- CSS “Compresses” CSS files to reduce file size (and save bandwidth)
- Removes extra spaces, new lines, comments, etc.
- Optimize for shorthands

```
.danger {  
  color: red;  
  font-size: 20px;  
  font-weight: bold;  
  font-family: Arial;  
}
```



```
.danger{color:red;font:700 20px Arial}
```



font-weight is 700 for bold

- Linter

- Performs syntax validation (like a compiler)
- Performs style and formatting analysis

# CSS Functions

- CSS is declarative, but it has some useful functions
- var()
  - Use a custom defined variable in place of a property value

```
:root { --main-bg-color: pink; }  
body { background-color: var(--main-bg-color); }
```

- url()
  - Use for properties that references a file (usually image)

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

- max(), min()
  - Selects the maximum or minimum of a set of values

```
width: max(20vw, 400px);
```



# CSS Animations

- [https://www.w3schools.com/css/css3\\_animations.asp](https://www.w3schools.com/css/css3_animations.asp)
- Allows animating HTML elements natively (without JS or Flash)

```
.loader {  
  margin: auto;  
  border: 40px solid lightgrey;  
  border-radius: 50%;  
  border-top: 40px solid orange;  
  width: 160px;  
  height: 160px;  
  /* short for name duration timing-function iteration-count */  
  animation: spinner 4s linear infinite;  
}  
@keyframes spinner {  
  0% { transform: rotate(0deg); }  
  100% { transform: rotate(360deg); }  
}
```

Properties at various points of the animation

# CSS Preprocessor

- A language on top of CSS that provides imperative programming features
- *Sass* Sass, **less** Less, *stylus* Stylus
- Can declare variables, create loops, support inheritance
  - Helps with writing concise and maintainable CSS
  - Especially useful for making powerful animations
- Interpreter (or compiler) translates preprocessor script into CSS
  - Usually done automatically as soon as you update the script

```
$font-stack: Helvetica, sans-serif;  
$primary-color: #333;  
body {  
  font: 100% $font-stack;  
  color: $primary-color;  
}
```



```
body {  
  font: 100% Helvetica, sans-serif;  
  color: #333;  
}
```

# Before you go

- Next week
  - Install Django on your local computer
  - <https://docs.djangoproject.com/en/4.1/topics/install/>
- Assignment 1
  - Due Sunday January 29<sup>th</sup>
- Project
  - We will assign you random partners if you have not chosen yet
    - Deadline is Friday January 20<sup>th</sup>
  - Sign up on Calendly to book an interview session for project grading
    - <https://calendly.com/csc309-2023s>

# Miscellaneous Properties

- list-style-type
  - The marker for list (unordered or ordered)
  - <https://developer.mozilla.org/en-US/docs/Web/CSS/list-style-type>
- z-index
  - Determines which element is on top when multiple elements overlap
  - Larger the better
- opacity
  - Transparency of an element; from 0.0 (invisible) to 1.0 (fully visible)
- transform
  - Allows for rotation, skewing, scaling, etc, of an element