Lecture 2:
Review of HTML and CSS

05-431/631 Software Structures for User Interfaces (SSUI)

Fall, 2022

Logistics

- PowerPoint format OK for slides?
- Anyone need transcripts for the video recordings?

What they are

- Html - Hypertext Markup Language
  - Describes the content and structure
    - And originally the look and behaviors
  - Declarative
  - oldest part: Invented by Tim Berners-Lee around 1991
- CSS - Cascading Style Sheets
  - Describe the look (“styles”)
  - hierarchical, reusable definitions
  - Declarative
  - Started around 1996
- JavaScript (JS)
  - Originally by Netscape, called “LiveScript” from about 1995
  - Renamed JavaScript for better publicity
  - Imperative, Object-oriented programming language
- HTML5 + CSS3 + JS around 2011
  - Sufficient to build almost any regular UI
  - No longer needed Flash, other plugins

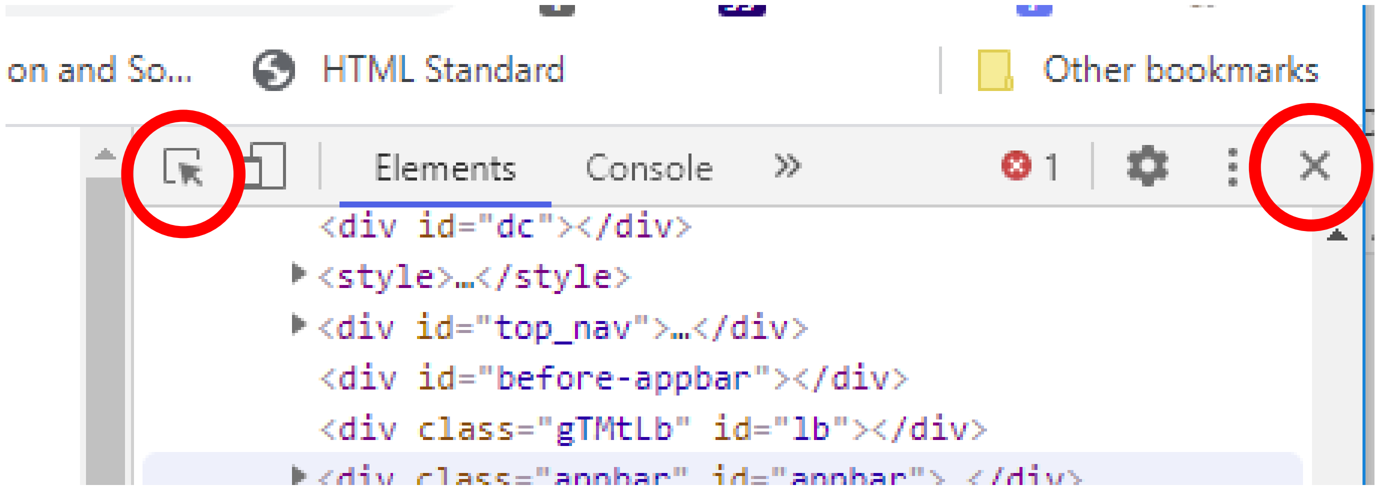
What I will cover

- Just a review
- Lots of details for each language not covered
- Some will be covered in Alex’s lab on Friday
- Excellent training and review sites online
  - See schedule for some recommendations
- Cover key structures and interactions here
  - You can lookup the details as needed
- Focus on the *differences* and how work together

Human-Computer Interaction Institute

Useful for debugging all of these

- Chrome debugger
  - Bring up with F12 or ^shift-I on Windows
  - Mac: Command+Option+I
- Console, Elements, Sources, Styles – most useful
- *Inspect* elements of the page
  - right-click the element and select **Inspect**. Or press Command + Option + C (Mac) or Control + Shift + C (**Windows**, Linux,**Chrome** OS).
  - Use “inspect” button
  - Also note close: “x”
    - Or F12 again
  - Highlights code or elements – wherever mouse is



```
<div id="dc"></div>
▸<style>…</style>
▸<div id="top_nav">…</div>
<div id="before-appbar"></div>
<div class="gTMtLb" id="lb"></div>
```

HTML

HTML - Syntax

- *tags* with `<xx>` end with `</xx>`: `<h1>big header</h1>`
- Content *not* in quotes
- No escape character, use `&qt;` for >, `&amp;` for &
- Multiple spaces, tabs and returns ignored, use `<p>` … `</p>` for paragraphs, `<br/>` for line breaks (`/>` since no closing tag), `&nbsp;` for each extra space
  - Single space often important, like `<a href="">lnk</a>.` ← no space before.
  - But don't use multiple spaces for formatting
- Tags with parameters: `<tag param1="value1" param2="value2">`
  - Values in quotes, even if numbers (which often have units)
  - Just spaces between
  - Examples: `<a href="SSUI-hw1.zip">` -- hyperlinks
    `<img src="HW1-products-quickview.png" width="19%" height="19%" alt="example products quickview page" />`
- Comments: `<!-- commented out -->`
- Names can contain hyphens: `<p style="font-family:verdana">`
- Capitals rarely matter `<li>` = `<LI>`, but do sometimes - case sensitive about URLs, IDs, etc.
- Browsers try hard to parse and display even if errors

Some important html elements

- <p>
- <ul> <ol> <li>
- <em>, <strong>
- <h1><h2>…
- <div>
- <span>
- <table> <tr> <th> <td>
- <image>
- <a href="“”>

HTML – important tags

- Top of file:
```
<html lang="en">
<head>
 …
 <link rel="stylesheet" href="style.css" />
 <title>Homeworks …</title>
</head>
<body>
```
  `<title>` -- important for bookmarks, icons, etc.
- `<h1>1st header</h1> … <h2>2nd level header</h2>`
- `<p>…</p>  <br/>`
- `<ul>` - unnumbered list `<li>list item</li> <li>2nd</li>…</ul>`
  - `<ol>` - numbered list, `<ol type="a">`
- `<img src= "picture URL"/>`
- `<a href="`*`hyperlink URL`*`">` -- may be a local or absolute reference
- `<div> … </div>` - block-level element, used extensively (“division”)
- `<span> … </span>` - text-level element, for a small number of words (or pictures)
  - Usually these have parameters of the CSS class

HTML Tables

- Used to be a key way to layout pages
  - Not appropriate any more – use div that are placed
- Still OK for tables, like the schedule

```
<table class="table table-bordered schedule-table">
         <thead class="thead">
           <tr>
             <th>Num</th>
             <th>Date</th>
             <th>Class Content</th>
           </tr>
         </thead>

         <tr>
           <td>1</td>
     ….
```

ID of element

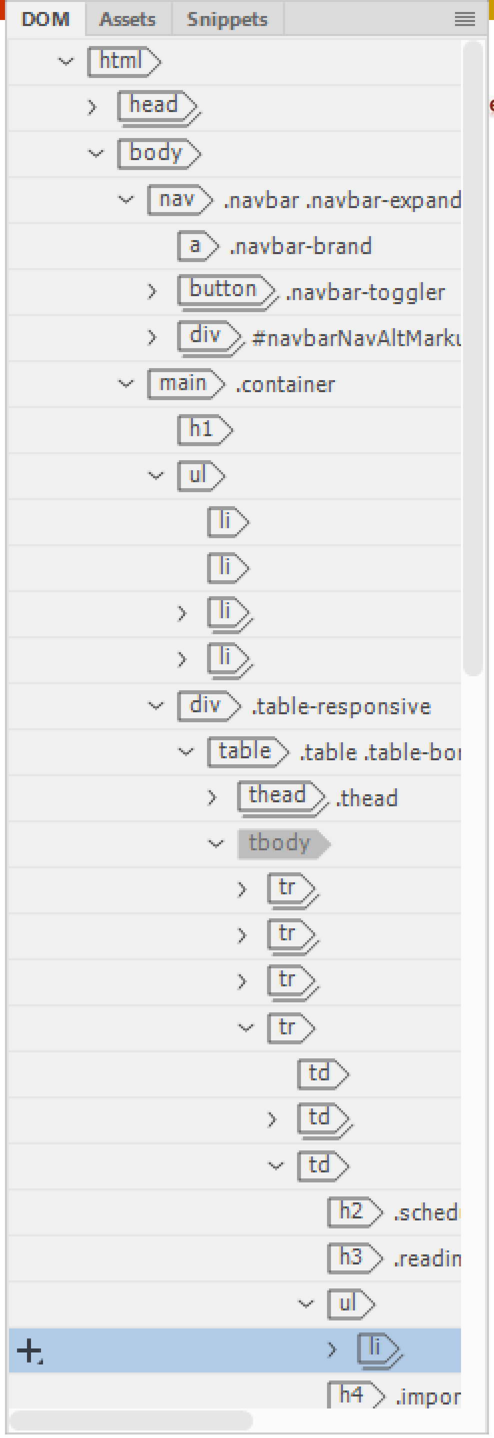
- Elements can have an ID
  - Must be *unique per page*
  - Case sensitive
  - Super useful!
- `<h2 id=“policies”>`
- Can be used as part of URL:
  - `<a href=“homeworks.html#policies">`
- Can be as reference for a style
  - Style only applies to this particular element
- Can be found in JavaScript:
  - `var policies = document.getElementById("policies");`

HTML input tags

- Usually wrapped in a <form>
- <input type="*value*">
  - Value can be button, checkbox, color (picker), date, …
  - `<input type="button" value="Click me" onclick="msg()">`
- Events and event handling in lecture 4
  - Any html object can have an onclick action, as in homework, e.g., from JavaScript:
  - `element.addEventListener("click", funcToCall);`

DOM

- Document Object Model
- Hierarchical structure of the web page document
- Used by renderer, CSS and JavaScript
- “Components”, “Containers”, “parent-child”
- Often surprisingly deep
- Can inspect with “Elements” of Chrome debugger



```
html
  head
  body
    nav .navbar .navbar-expand
      a .navbar-brand
      button .navbar-toggler
      div #navbarNavAltMarku
    main .container
      h1
      ul
        li
        li
        li
        li
      div .table-responsive
        table .table .table-bo
          thead .thead
          tbody
            tr
            tr
            tr
            tr
              td
              td
              td
                h2 .sched
                h3 .readin
                ul
                  li
                h4 .impor
```

CSS

CSS syntax

-
  ```
  selector {
      prop1 : value1 ; /* comment */
      prop2 : value2 ;
  }
  ```
- Completely different than html (closer to JS)
  - Bracketed with { }
  - No quotes for values
  - ":" for assignment
  - separated by ";"
  - Comments as /* comment */
- But names *still* can contain "-" : `font-size`

Options for “selector”

- What this style is connected to
- (1) Name by itself = html tag:

```
p {
  color: red;
  text-align: center;
}
```

- (2) `#ID` for referencing IDs on the page
  - Format just that one item

```
#policies {
    text-align: right;
}
```

CSS Classes

- (3) can select a CSS *class*
  - Note – not related to JS “class”
- CSS class – name cannot start with a number
- Reference in CSS file by starting with a period

```
.slides {
  font-size: 1.25rem;
  font-weight: normal;
  font-style: normal;
  margin-top: 12px;
  margin-bottom: 12px;
}
```

  - Reference in html (*no period*): `<h3 class="slides">`
  - Can reference more than one
    - Separate with a space: `<h3 class="slides homeworks“>`

Selectors in CSS File

- Can group more than one *using comma*

```
h1, h2, p {
  text-align: center;
  color: red;
}
```

- Space used for *descendent* (anywhere down the hierarchy) = “”
  - Any <p> inside a <div>:

```
div p {
  background-color: yellow;
}
```

- Greater than “>” for <u>immediate</u> *child* only

```
div > p {
  background-color: yellow;
}

<div>
  <p>Paragraph 1 in the div.</p>

  <section>
      <p>Paragraph 3 in the div.</p> <!-- not Child but is Descendant -->
  </section>
</div>
```

More on selectors

- Pseudo-classes for built-in states of elements
  - :hover = while mouse is over it
  - :link = unvisited hyperlink
  - :visited = visited hyperlink
  - :active = while mouse is pressed over it
  - :root = top document, usually first in the CSS file
- Designated with colon :

```
.mybutton:hover {
  color: hotpink;
}
```

- Can be combined, e.g., with class

```
a.mylinkclass:hover {
  color: #ff0000;
}
```

  - Only links marked with class `mylinkclass` will have hover
- Specific order: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective! a:active MUST come after a:hover

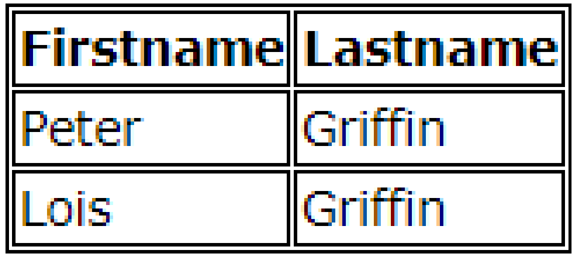
`a:hover` MUST come after `a:link` and `a:visited` in the CSS definition in order to be effective! `a:active` MUST come after `a:hover`

Some units of measure for Values

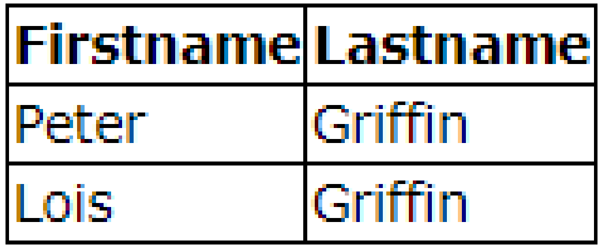
- `in` = inches, 1in = 96px = `2.54cm`
- `px` = pixels, 1px = 1/96th of 1in
- `pt` = points, 1pt = 1/72 of 1in
- `%` = percent, relative to the parent element, 50%
- `rem` = relative to font-size of the root element
  `font-size: 1.25rem;`
- `#a3` = hexadecimal (base 16) = 163 ($\underline{1010}0011$)

Some useful properties

- `color` – text or foreground color
  - `background-color` – behind the text (or `background-image`, etc.)
  - Values = `rgb(23, 137, 179);` // out of 0..255
    - `#1789b3` → `#17 = 16*1 + 7 = 23` … `#b3 = 11*16+3=179`
    - `rgba(23, 137, 179, 0.05);`
    - https://www.w3schools.com/colors/colors_picker.asp?color=1789b3
- `border-style: dotted, dashed, none,` …
  - `border-bottom-style`
- `border-width: 5px;`
- `border-radius` – rounded corners
- `border-collapse: collapse` – esp. for tables



Firstname	Lastname
Peter	Griffin
Lois	Griffin



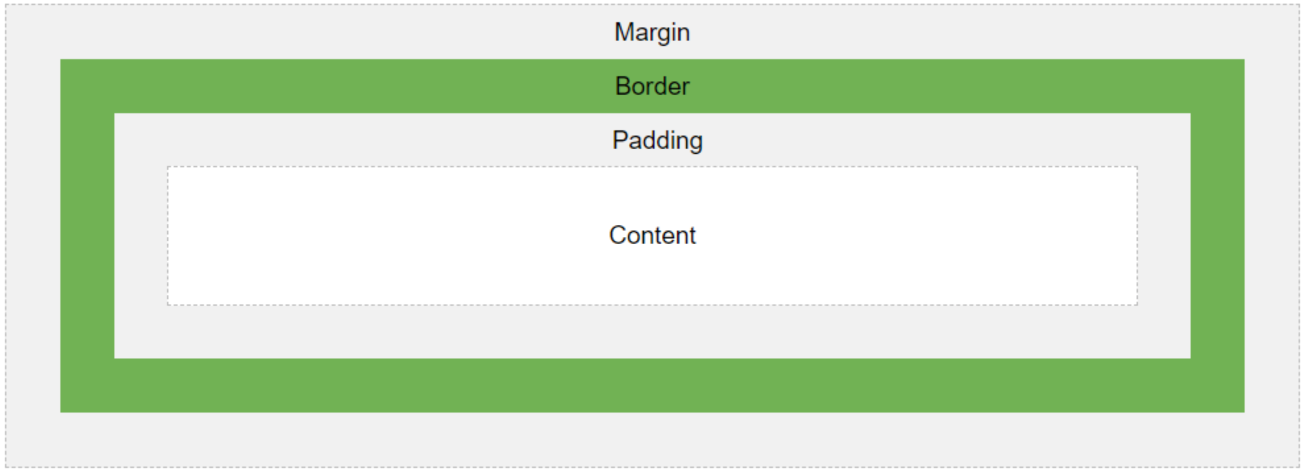
Firstname	Lastname
Peter	Griffin
Lois	Griffin

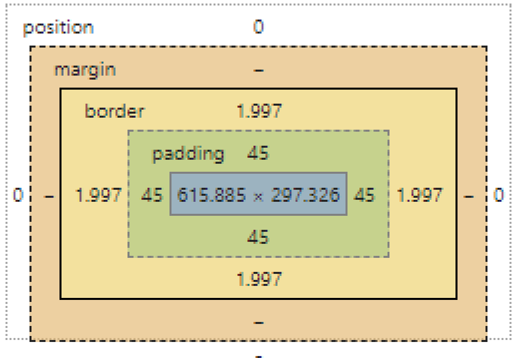
Positions and size

- `left`, `top`, `width`, `height`, `right`, `bottom`
- `position`:
  - `static` – flows with other elements, default
  - `relative` – to its parent
  - `fixed` - relative to the viewport
  - `absolute` – based on container that has a specific position (any that isn't static)
  - `sticky` – scrolls then sticks
- `Float`: `left` or `right` or `none`
  - Often used for pictures
  - Text fills around it

Margins & Padding

- `margin` – all 4 margins
- `margin-top, margin-bottom, margin-right, margin-left`
- What about elements next to each other?
  - Margin collapse = takes max of `margin-bottom` of first and `margin-top` of lower
  - Right and left just add
- `padding` – outside content, but inside margins
- Also available at bottom of Chrome inspector “Styles” tab





Text

- `text-align: center` or `left` or `right` or `justify`
- `vertical-align: top` or `middle` or `bottom` – useful in table cells
- `font-family: "Times New Roman", Times, serif;`
- `font-style: normal` or `italic`
- `font-weight: normal` or `bold`
- `font-size: 14px` or other units
- `text-decoration: none;` - often used to remove underline for links

Special Properties

- `display: none;` -- remove the element, often used when dynamically shown or hidden
- `visibility: hidden` – remove element but leave its room

- `display: inline;` -- override normal linefeed, e.g., to make <ul> lists horizontal

- `display: flex;` - flexbox: put in a row or column
  - `justify-content: space-between;` - spread out to fill the row
  - `flex-wrap: wrap;` - multiple rows if don't fit

- **`transition: 0.5s;` - animate change over ½ second**

Adjusting to size of screen

- For *responsive* web pages, that adjust based on size or orientation of the screen

```
.nav-item.divider { …
}
@media (max-width: 767px) {
  .nav-item.divider {
    display: none;
  }
}
```

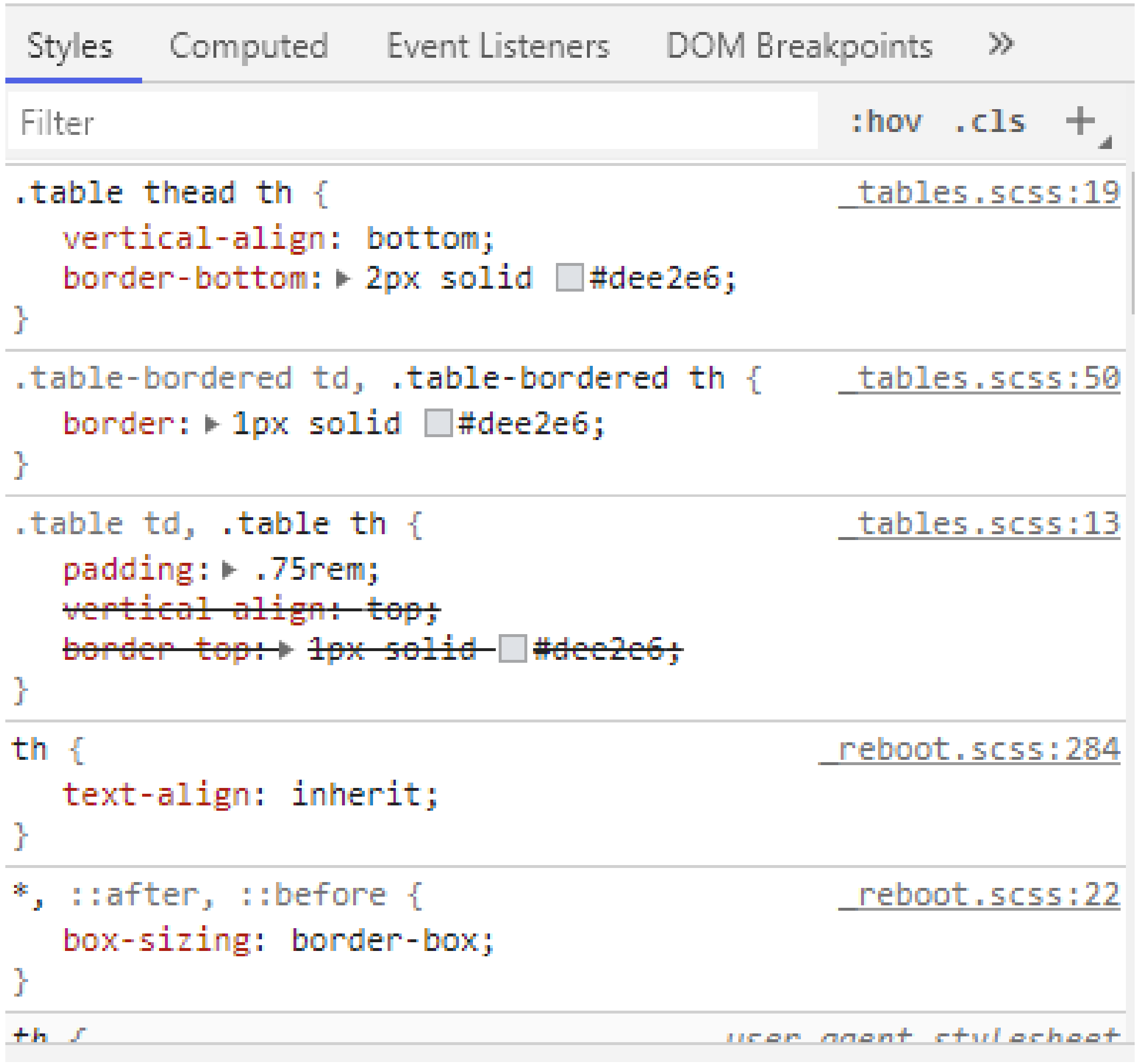
  - Only used when width of browser window > 767px
  - Overrides definitions just above, or adds to them
- Can have multiple with different ranges
- *(not required for HW1!)*

Cascading

- Properties overridden
  - E.g., vertical-align set in multiple places
  - Inspector shows which ones in force
- Source order – latest wins
- Tag type < Class < ID
  - More specific wins (right most)
- Inherited from parent→ child, e.g.,

  `body { color: blue; }`

  will be inherited unless overridden
  - Note: “inheritance” here is by component, not type
- Can overrule with `!important;`



```
.table thead th {                                  tables.scss:19
    vertical-align: bottom;
    border-bottom: ▸ 2px solid ▢#dee2e6;
}
.table-bordered td, .table-bordered th {           tables.scss:50
    border: ▸ 1px solid ▢#dee2e6;
}
.table td, .table th {                             tables.scss:13
    padding: ▸ .75rem;
    ~~vertical-align: top;~~
    ~~border-top: ▸ 1px solid ▢#dee2e6;~~
}
th {                                               reboot.scss:284
    text-align: inherit;
}
*, ::after, ::before {                             reboot.scss:22
    box-sizing: border-box;
}
```

Putting them together

- Html
  - Main file is `*/index.html`
  - Typically, one html file per web page
  - Each .html file references the needed JS and CSS files at the top
- Put JavaScript into own file
  - Often `index.js` or `main.js` unless a shared library
  - Or directly in the html file `<script> … </script>`
  - (Later with React-JS will be different)
- CSS also in its own file
  - Often `style.css`
  - Also can be in the html file `<style> … </style>`