Important Information

- Masks are **required** in the classroom
  - No eating or drinking in Rice 130
- Lectures will be streamed on Zoom and posted to Panopto
- If you feel unwell, please watch remotely

Programming Languages for Web Applications

HyperText Markup Language Cascading Style Sheets

Robbie Hott with content from Praphamontripong, Soffa

Administrativa

- Sprint 1 due today
- Homework 1 due Feb 15

Tim Berners-Lee

"HyperText is a way to link and access information of various kinds as a web of nodes in which the user can browse at will. Potentially, HyperText provides a single user-interface to many large classes of stored information, such as reports, notes, data-bases, computer documentation and on-line systems help. We propose the implementation of a simple scheme to incorporate several different servers of machine-stored information already available at CERN, including an analysis of the requirements for information access needs by experiments... A program which provides access to the hypertext world we call a browser. "

— T. Berners-Lee, R. Cailliau, 12 November 1990, CERN

The Basic Idea

The Tardis Tales

A collection of Doctor Who themed fan fiction, centered around the 10th and 11th doctors and their companions.

Story 1 – The two doctors
…

Your Content

+

<html>
<body>
<p>
<h1>
<nav>
<h2>
<a>
<table>
<b> <u> <i>
<blink>
<img>

HTML Markup
(structure)

=

The Tardis Tales

A collection of Doctor Who themed fan fiction, centered around the **10th** and **11th** doctors and their companions

- Story 1 – The two doctors

…

Your Website

The Basic Idea – Take Two

The Tardis Tales

A collection of Doctor Who themed fan fiction, centered around the 10th and 11th doctors and their companions.

Story 1 – The two doctors
…

\+

```
<html>
  <body>
<p>      <nav>
    <h1>
<h2>      <a>
    <table>
<b> <u> <i>
 <blink>
       <img>
```

\+

```
body{
 background-color: #3578cd;
 color: #fff;
}

p{
 font-family: Times, serif;
 color: #fff;
}

h1 {
 font-family: Arial;
}

…
```

=

The Tardis Tales

A collection of Doctor Who themed fan fiction, centered around the **10th** and **11th** doctors and their companions

- Story 1 – The two doctors

…

Your Content | HTML Markup (structure) | CSS (styling) | Your Website

What is HTML?

HyperText Markup Language

- Language for describing **structure of a document**
- A **text file** containing **markup tags** (or elements)
- Markup tags tell the web browser *how* to display the page
- HTML provides hierarchy of elements
- Can be created using a simple text editor, HTML editor (WYSIWYG), or IDE

HTML Elements / Tags

```
<a href="page.html">This is a link.</a>
```

attribute name

attribute value

Start an anchor element

Opening tag begins an HTML element.

Opening tags **must** have a corresponding closing tag.

Set the link target location (href) to page.html

HTML **attributes** are name-value pairs that provide additional information or context about the contents of an element.

End the anchor element

Closing tag ends an HTML element. All content between the tags *and the tags themselves* comprise an HTML element.

HTML Elements / Tags

```
<img src="myimage.png" alt="Tardis picture" />
```

Self-closing elements

Some HTML elements are self-closing and include a built-in closing tag (/>)

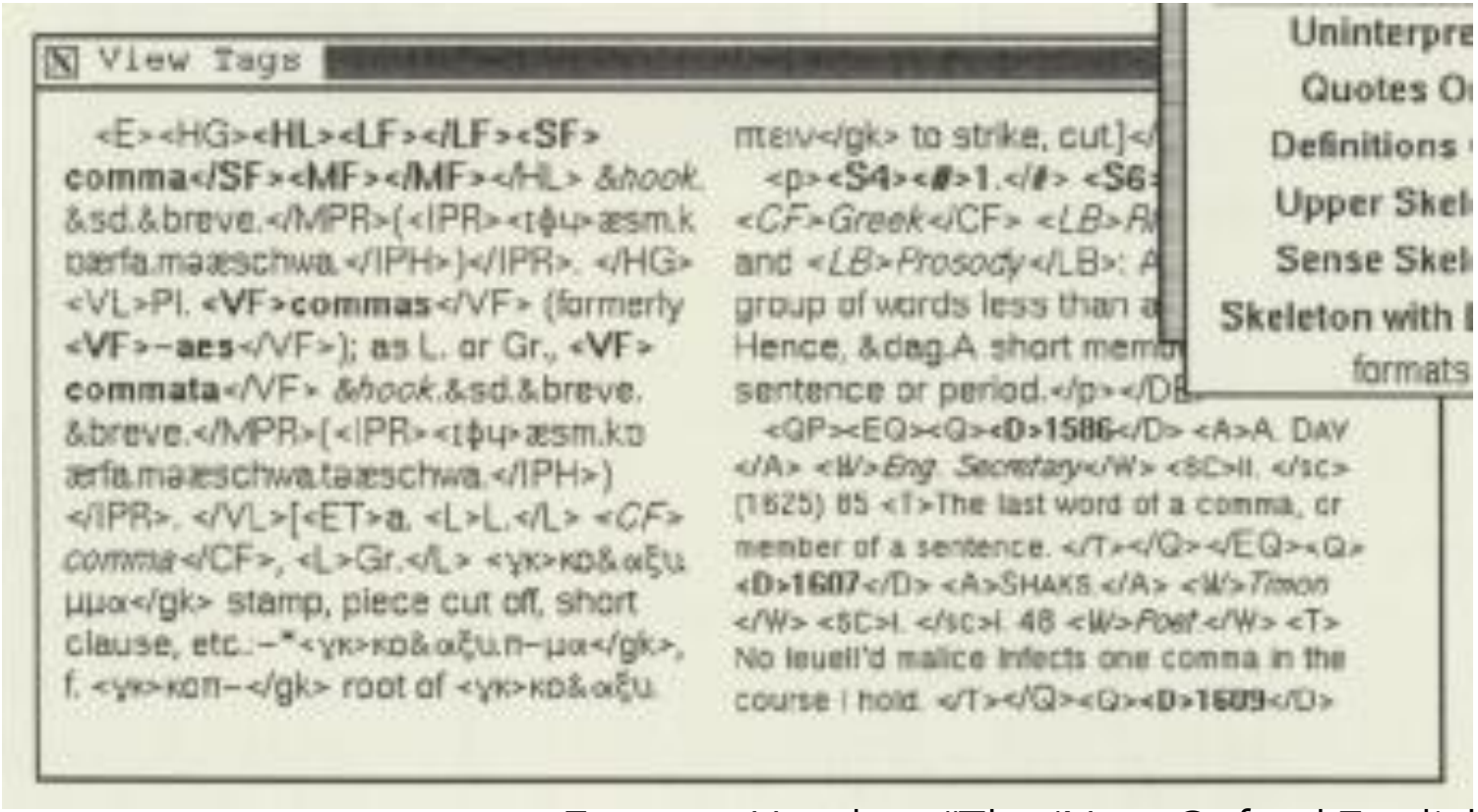
History - SGML

Standard Generalized Markup Language (SGML)

- Descended from IBM's Generalized Markup Language (GML) developed in 1960s
- Designed for machine-readable, sharable documents
- Became an international standard in 1986
  - ISO 8879:1986 - Information processing — Text and office systems — Standard Generalized Markup Language (SGML)
- Reworked into XML (a subset of SGML)
  - We mostly see XML today, SGML is rarely used

https://en.wikipedia.org/wiki/Standard_Generalized_Markup_Language

History - SGML



Fawcett, Heather. "The 'New Oxford English Dictionary' Project." *Technical Communication* 40, no. 3 (1993): 379–82. http://www.jstor.org/stable/43091009.

History - HTML

- Late 1980s: Tim Berners-Lee created first HTML version
  - Intended to be an application of SGML, but developed independently and in parallel
- 1995: HTML 2.0 published as standard with RFC 1866
- 1997: HTML 3.0 published as W3C Recommendation
- 1997: HTML 4.0 published as W3C Recommendation
  - Standardized most modern HTML elements / tags
  - Three flavors
    - Strict: deprecated elements are forbidden
    - Transitional: deprecated elements are allowed
    - Frameset: only frame-related elements allowed
  - Encouraged use of CSS for styling elements over HTML attributes
  - Fully conforms to ISO 8879 (SGML)

https://en.wikipedia.org/wiki/HTML

W3C – World Wide Web Consortium

- Website: w3.org
- Director: Tim Berners-Lee
- International community that develops web standards
  - See: https://www.w3.org/standards/webdesign/
- Provides services such as HTML Validator
  - https://validator.w3.org/

History - HTML

- Late 1980s: Tim Berners-Lee created first HTML version
  - Intended to be an application of SGML, but developed independently and in parallel
- 1995: HTML 2.0 published as standard with RFC 1866
- 1997: HTML 3.0 published as W3C Recommendation
- 1997: HTML 4.0 published as W3C Recommendation
  - Standardized most modern HTML elements / tags
  - Three flavors
    - Strict: deprecated elements are forbidden
    - Transitional: deprecated elements are allowed
    - Frameset: only frame-related elements allowed
  - Encouraged use of CSS for styling elements over HTML attributes
  - Fully conforms to ISO 8879 (SGML)

https://en.wikipedia.org/wiki/HTML

History - HTML

- 2000: XHTML 1.0
  - Imposed stricter rules on HTML format
    - E.g., elements needed closing tag, attribute names in lowercase
- 2014: HTML5 published as W3C recommendation
  - New features for capturing more semantic information and declarative description of behavior
    - E.g., input constraints, new tags that explain purpose of content
  - Important changes to DOM
  - HTML 5.3 published as W3C recommendation in 2017

https://en.wikipedia.org/wiki/HTML

History - HTML

- 2021: HTML5.3 retired by W3C
- HTML Living Standard: https://html.spec.whatwg.org
  - Web Hypertext Application Technology Working Group (WHATWG)
  - Steering Committee: Apple, Mozilla, Google, and Microsoft
  - No longer versioning HTML

https://en.wikipedia.org/wiki/HTML

HTML Contents

Type	Description	Example
Metadata	Content hosted in the head of an HTML document. Doesn't appear in the web page but is used to describe a webpage and its relationships to other external resources	`<meta name="viewport" content="width=device-width, initial-scale=1">`

Flow	Text and all elements that can appear as content in the body of an HTML document	`<body>`
		`  <h1>Heading</h1>`
		`  <p>Some content…</p>`
		`</body>`

Sectioning	Used to structure the content of a web page and to help with layout	`<section class="highlight col">`
		`  Some content …`
		`</section>`

HTML Contents

Type	Description	Example

Phrasing	Elements for marking up content within a paragraph element such as text and typography	`<p>`
		`  <b>Emphasized text</b>`
		`  and some normal text`
		`</p>`

Heading	Elements used to define the headings of a section of an HTML document. The elements `h1-6` represent headings with `h1` having the highest ranking	`<h1>Main heading</h1>` `<h2>Sub-heading</h2>`

Embedded	Embedded content includes media, such as video, audio, and images	`<img src="media/monster.png"`
		`alt="A cute monster image"`
		`width="80%" />`

Interactive	Elements that a user can interact with such as media elements with controls, form inputs, buttons, and links	`<input type="text" name="username"`
		`required />`
		`<input type="password" name="pwd"`
		`required />`

HTML Example

Use HTML5 standard mode

Interpret bytes as UTF-8 characters
Includes both ASCII and international characters

HTML content

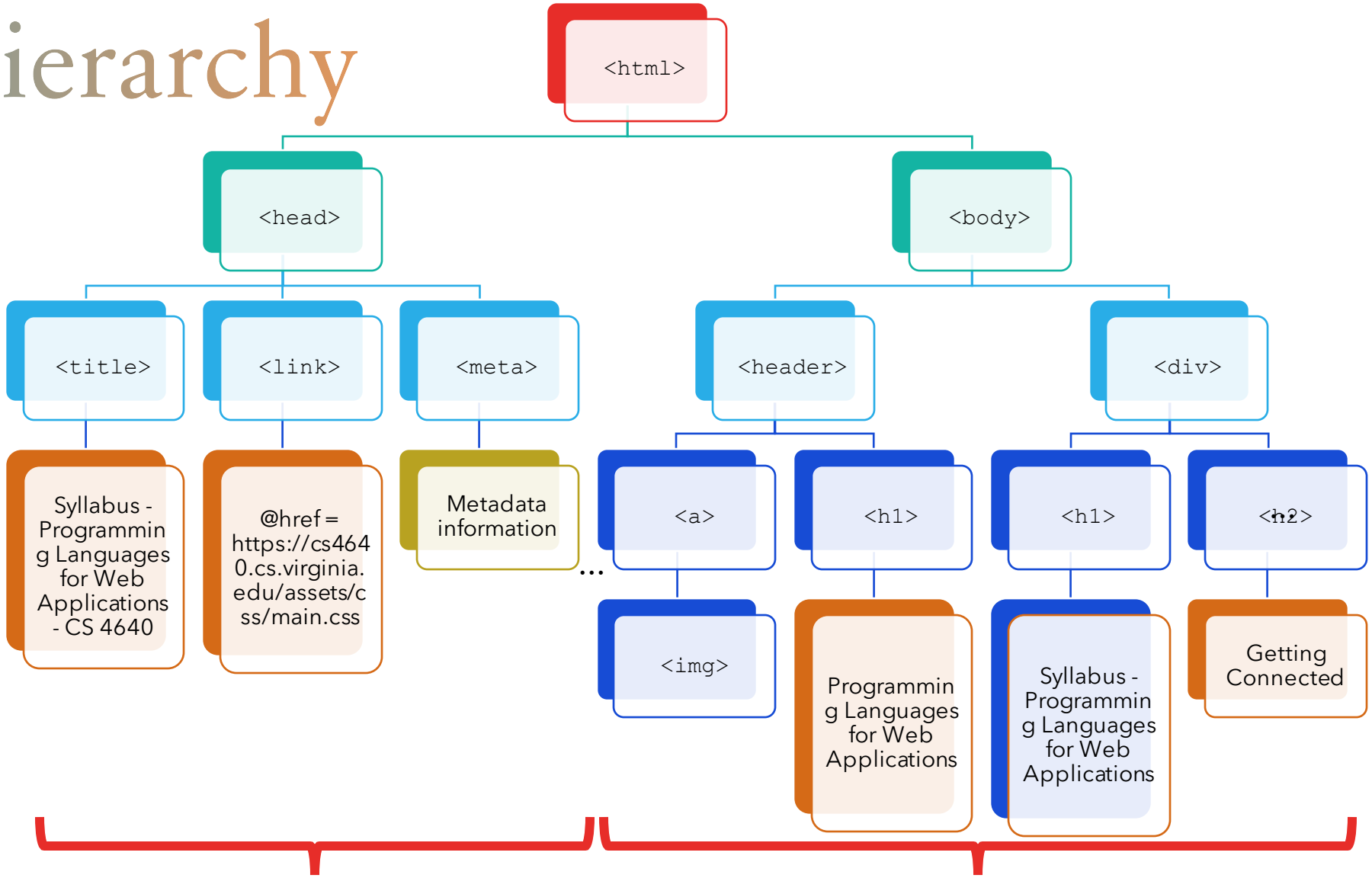
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
    <!- this is a comment -->
    <!- anything in the head section is not rendered on the screen -->
  </head>
  <body>
    <!- anything in the body section is rendered on the screen -->

    <p lang="en-us">This is a paragraph in English</p>
    <a href="https://cs4640.cs.virginia.edu/schedule/">
      CS4640 Schedule
    </a>
  </body>
</html>
```

Title
Used by browser for title bar or tab

Header
Information about the page

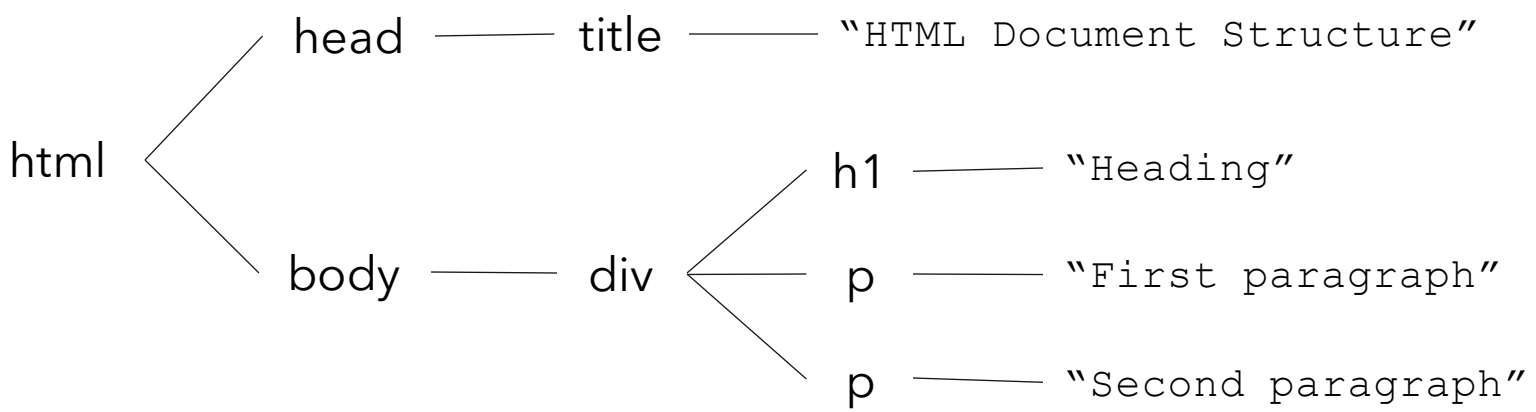
Document content

HTML Hierarchy



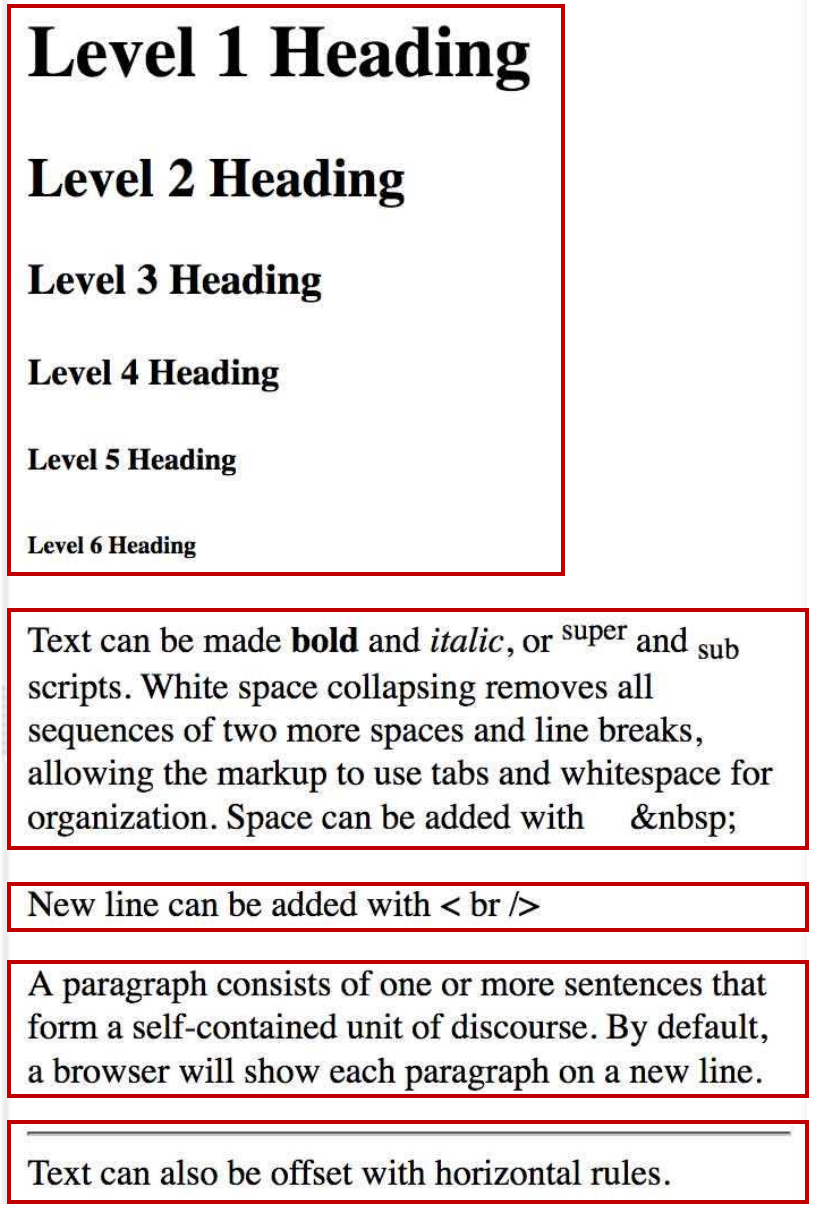
HTML Hierarchy - Structure

```
<html>
  <head>
    <title>HTML Document Structure</title>
  </head>
  <body>
    <div>
      <h1>Heading</h1>
      <p>First paragraph</p>
      <p>Second paragraph</p>
    </div>
  </body>
</html>
```



Body Text

```
<!doctype html>
<html>
<head></head>
<body>

<h1>Level 1 Heading</h1>
<h2>Level 2 Heading</h2>
<h3>Level 3 Heading</h3>
<h4>Level 4 Heading</h4>
<h5>Level 5 Heading</h5>
<h6>Level 6 Heading</h6>

Text can be made <b>bold</b> and <i>italic</i>,
or <sup>super</sup> and <sub>sub</sub> scripts.
White space collapsing removes all sequences of
two more spaces and line breaks, allowing
the markup to use tabs and whitespace for
organization.
Space can be added with     &amp;nbsp;
<br />
<br />New line can be added with &lt; br /&gt;

<p>A paragraph consists of one or more sentences
   that form a self-contained unit of discourse.
   By default, a browser will show each paragraph
   on a new line.
</p>

<hr />
Text can also be offset with horizontal rules.

</body>
</html>

```



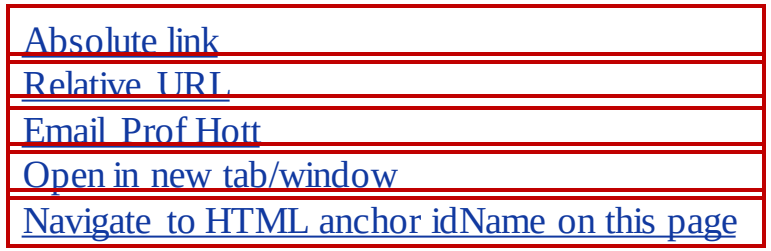
Semantic Markup Tags

- Tags that can be used to denote the meaning of specific content

	Examples
	An element that has importance
<blockquote>	An element that is a long quote
<q>	A short quote inline in paragraph
<abbr>	Abbreviation
<cite>	Reference to a work
<dfn>	The definition of a term
<address>	Contact information
<ins>	Content that is inserted
<del>	Content that is deleted

Links

```
<!doctype html>
<html>
<head>
</head>
<body>
  <a href="https://cs4640.cs.virginia.edu">Absolute link</a>
  <br />
  <a href="schedule.html">Relative URL</a>
  <br />
  <a href="mailto:jrhott@virginia.edu">Email Prof Hott</a>
  <br />
  <a href="schedule.html" target=" blank">Open in new tab/window</a>
  <br />
  <a href="#idName">Navigate to HTML anchor idName on this page</a>
</body>
</html>
```



Embeds: Image, Audio, Video

- HTML has built-in support for:
  - Images: `<img>` - .jpg, .png, .gif, .svg, etc
  - Audio: `<audio>` - .mp3, webm
  - Videos: `<video>` - .mp4, webm
- Always use an `alt` attribute on images to ensure accessibility

```
<img src="tardis.png" alt="Tardis picture" />
```

Embeds: Image, Audio, Video

- Important attributes for `<video>` elements
  - `src` - location of video
  - `autoplay` - tells browser to start play
  - `controls` - show the default controls
  - `loop` - loop the video
  - `muted` - mutes the audio from the video

```
<video width="400" controls>
    <source src="doctor_who.mp4" type="video/mp4" />
    Your browser does not support HTML5 video.
</video>
```

Tables

```
1  <!doctype html>
2  <html>
3  <head>
4    <title>Example: Table</title>
5  </head>
6  <body>
7    <table border="2" cellspacing="2" bgcolor="lightyellow" width="70%"
         align="center">
8      <tr>
9        <th> </th>
10       <th>Monday</th>
11       <th>Tuesday</th>
12       <th>Thursday</th>
13     </tr>
14     <tr>
15       <th>1pm-2pm</th>
16       <td rowspan="2">Intro to Programming</td>
17       <td>Calculus</td>
18       <td> </td>
19     </tr>
20     <tr>
21       <th>2pm-3pm</th>
22       <td> </td>  <!-- why   here ? -->
23       <td>Physics</td>
24     </tr>
25   </table>
26 </body>
27 </html>
```

	Monday	Tuesday	Thursday
1pm-2pm	Intro to Programming	Calculus	
2pm-3pm			Physics

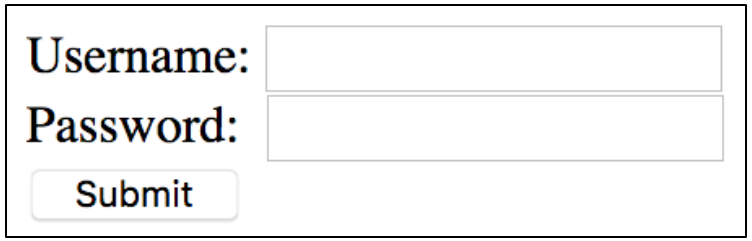
rowspan

Forms

Send form data to cs4640.cs.virginia.edu/formHandler.php

Action attribute should be omitted if not using form to submit data

```
<form action="https://cs4640.cs.virginia.edu/formHandler.php" method="post" >
   Username: <input type="text" name="username" value="blank" /> <br />
   Password: <input type="password" name="pwd" /> <br />
   <input type="submit" value="Submit" />
</form>
```



Transfer method

Method attribute specifies how data is transmitted to server.

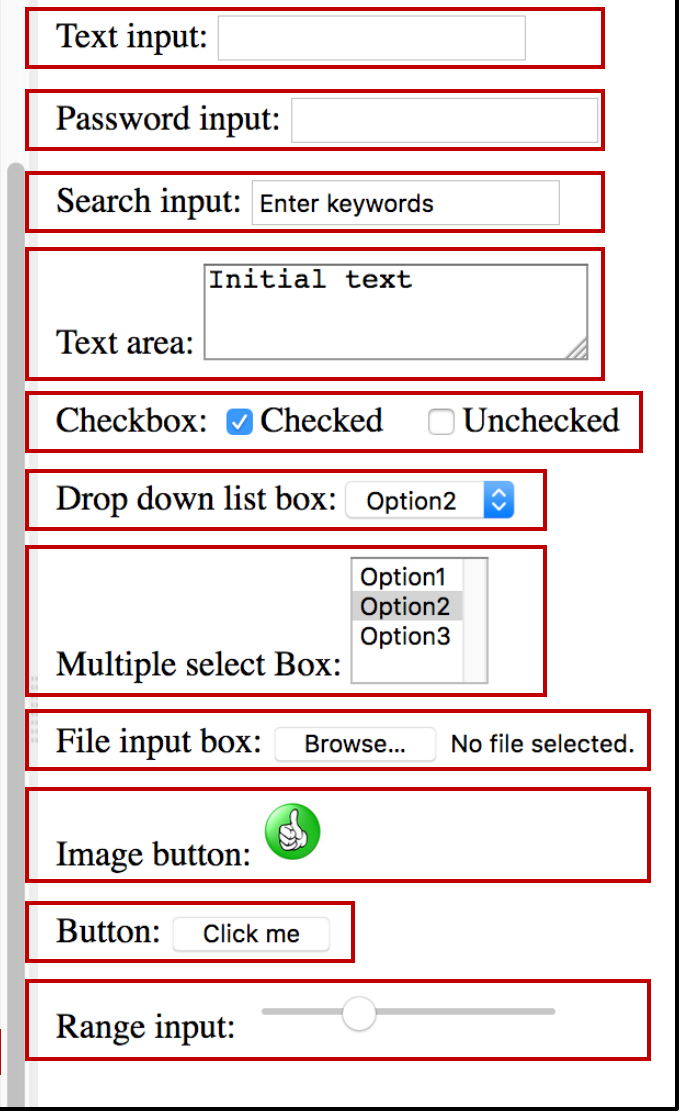
method="get" sends data appended to URL.

method="post" sends data as an HTML document

- Elements in a form are submitted to the server.
- A form may (or may not) have controls.

Controls

```
<p>Text input: <input type="text" maxlength="8" /></p>
<p>Password input: <input type="password" /></p>
<p>Search input: <input type="search" value="Enter keywords" /></p>
<p>Text area: <textarea>Initial text</textarea></p>
<p>Checkbox:
  <input type="checkbox" checked="checked" />Checked   
  <input type="checkbox" />Unchecked
</p>
<p>Drop down list box:
  <select>
    <option>Option1</option>
    <option selected>Option2</option>
    <option>Option3</option>
  </select>
</p>
<p>Multiple select Box:
  <select multiple>
    <option>Option1</option>
    <option selected>Option2</option>
    <option>Option3</option>
  </select>
</p>
<p>File input box: <input type="file" /></p>
<p>
  Image button: <input type="image"
  src="http://www.cs.virginia.edu/~up3f/cs4640/images/thumb-up.jpg"
  width="30" />
</p>
<p>Button: <button>Click me</button></p>
<p>Range input: <input type="range" min="0" max="100" step="10" value="30" />
</p>
```

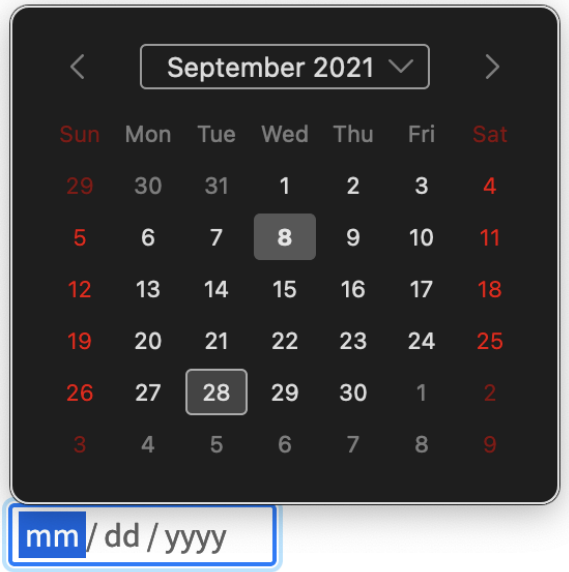


Specialized Controls

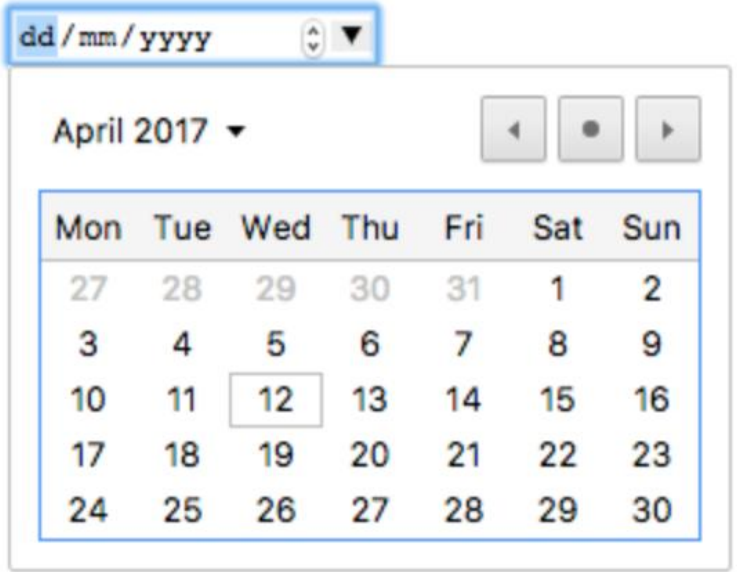
- `date`: may appear differently depending on browser or operating system

```
<input type="date" />
```

Firefox



Chrome



Specialized Controls

- `time`: may appear differently depending on browser or operating system

```
<input type="time" />
```

Firefox

Chrome

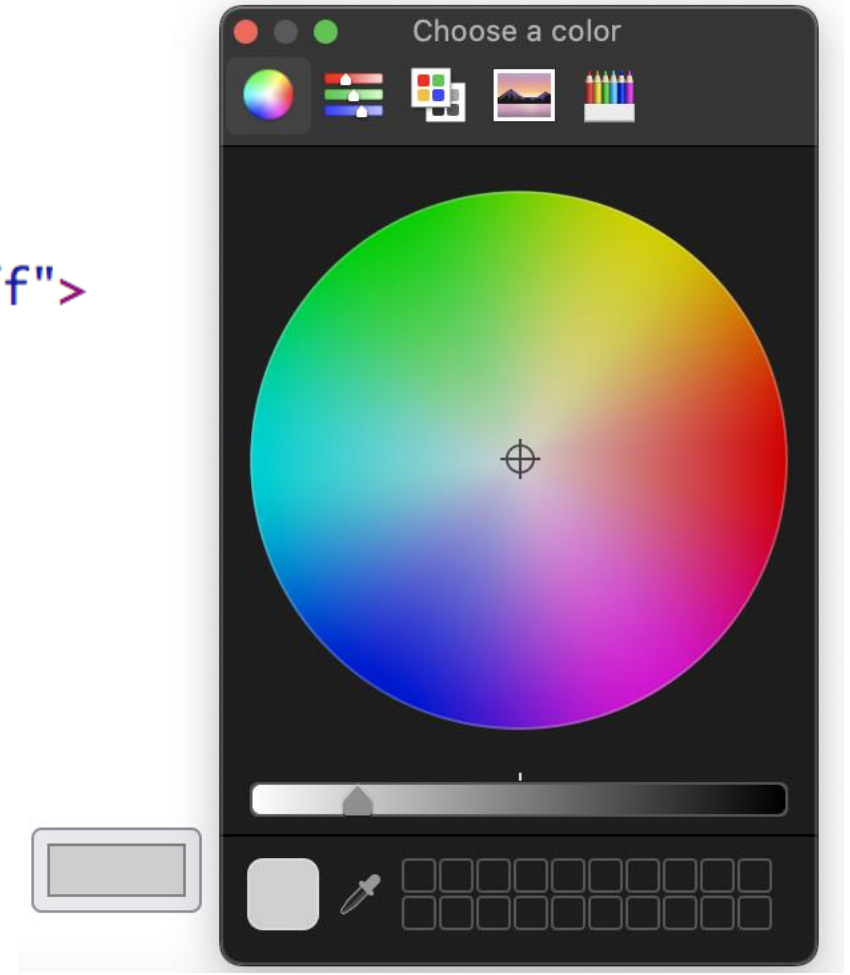
- `number`: can specify min and max values

```
<input type="number" min="1" max="12" />
```

Specialized Controls

- `color`: allow the user to choose colors

```
<input type="color" name="favcolor" value="#00ffff">
```



Labeling Inputs

- Placeholder suggestions can be given inside text boxes

```
<p>Input box: <input type="text" placeholder="Enter keyword" /></p>
```

Input box: Enter keyword

Disappear after the user begins typing

Input box: a

- Labels **should** be attached to inputs (accessibility)

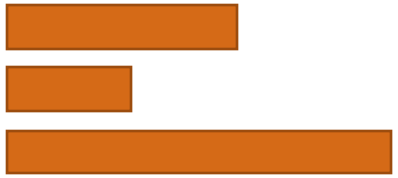
```
<p><label>
    Label on input box: <input type="text" placeholder="Enter keyword" />
  </label>
</p>
```

Label on input box: Enter keyword

Block and Inline Elements

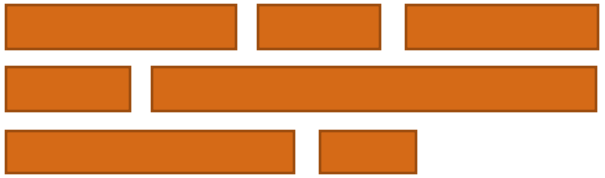
Block elements

- Appear on a new line
- Example:
  - `<h1>`
  - `<p>`
  - `<li>`
  - `<table>`
  - `<form>`
  - `<div>`



Inline elements

- Appear on the same line
- Example:
  - `<a>`
  - `<b>`
  - `<em>`
  - `<input>`
  - `<img>`
  - `<span>`



Validating Inputs (HTML)

Certain inputs will display errors on invalid input immediately, making it easier for users to fix errors

- Check that input is a valid **email**

```
<label>Email: <input type="email" /></label>
```

Email:

- Check that input is a valid **URL**

```
<label>URL: <input type="url" /></label>
```

URL:

- Constrain input to be at most **max length**

```
<label>Enter a username up to 10 characters:
  <input type="text" maxlength="10" />
</label>
```

Enter a username up to 10 characters:

Validating Inputs (HTML)

Certain inputs will display errors on invalid input immediately, making it easier for users to fix errors

- Check that input matches **regex pattern**

```
<label>Would you like coffee or tea?
  <input type="text" pattern="coffee|tea" />
</label>
```

Would you like coffee or tea?

- **Prevent** all edits

```
<label>Readonly text:
  <input type="text" readonly />
</label>
```

Readonly text:

Grouping Elements

- Elements can be grouped with generic or semantic elements
  - Creates a “parent” or “container” for a set of child elements
  - Useful for styling elements together or referencing elements in scripts
- Generic elements:
  - `<div>` : generic block element
  - `<span>` : generic inline element

Grouping Elements

- Semantic block layout elements are associated with meaning
  - Popular examples:
    - `<header>`, `<footer>`, `<nav>`, `<article>`, `<aside>`, `<section>`, `<figcaption>`

```
<!doctype html>
<html>
<head>
  <title>Example: Grouping Elements</title>
</head>
<body>

  <header>
    <h1>How to Get an A+</h1>
    <nav>...</nav>
  </header>
  <article>
    <section>
      <h3>Practice</h3>
      <p>When there are practice problems, ...</p>
    </section>
    <aside>
      <h4>Useful Links</h4>
      <a href="http://www.pythontutor.com/javascript.html">Javascript Tutor</a>
    </aside>
  </article>

</body>
</html>
```

Best Practices: Styling HTML

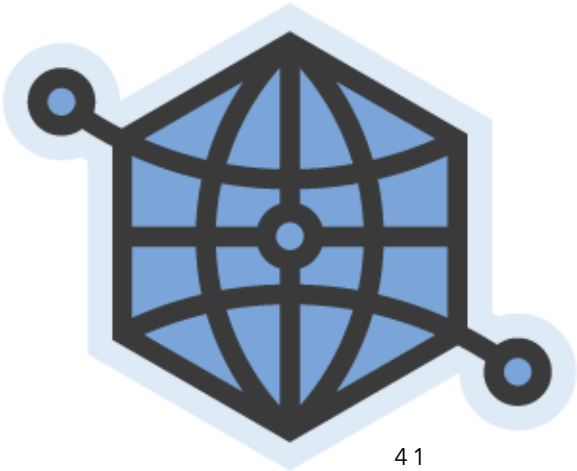
- Tags
  - Use lowercase for names
  - Use indentation to reflect hierarchy
  - **Always** close tags
- Use attribute `name="value"` format for attributes
- Use blank lines to break up documents into closely connected regions
- Use comments to describe purpose of regions

Best Practices: HTML

- Use specialized controls or input validation where applicable
- Use label or placeholder for labeling controls
- Use alt to make images accessible

OpenGraph Metadata

- Metadata to interconnect website more deeply in social graph
- Used by Facebook, Twitter, Outlook, and others
- Important elements:
  - `og:title` - title of the object
  - `og:type` - type of the object
  - `og:image` - an image that represents the object
  - `og:url` - url to the object
  - `og:description` - short description of the object
  - `og:site_name` - the larger site name



HTML Demo & Activity

Cascading Style Sheets

Cascading Style Sheets (CSS)

- Language used to describe the presentation of a web page
- Designed to **separate concerns**
  - HTML: content and structure of the document
  - CSS: look and feel (style and layout)
- Separating the visual presentation (CSS) from the structure (HTML) increases readability and maintainability
  - Reusing style across elements and pages
  - Update HTML without needing to update styles

Introducing CSS in HTML

Three ways to add CSS to HTML

- Inline – specify the style properties in the opening tag of an element
  - Applies to *that element* only
- Document-level – specify style in the document `<head>`
- External-level – specify style in a separate `.css` file, link to the file in the document `<head>`

Inline CSS

- Key-value properties are added to the style attribute

```
<tag style="property1:value1; property2:value2; ..."> ... </tag>
```

- For example:

```
<p style="text-align:center; color:red; font-weight:bold">
   This is a centered paragraph of bold red text.
</p>
```

Document-Level CSS

- Style specifications are added to <head> in the <style> tag
  - Body of the <style> tag is raw CSS format

```
<head>
   <style>
   p {
      text-align: center;
      color: red;
      font-weight: bold;
   }
   ...
   </style>
</head>
<body>
   <p>This is a centered paragraph of bold red text.</p>
</body>
```

External-Level CSS

- CSS written in separate file, included by HTML

file.html

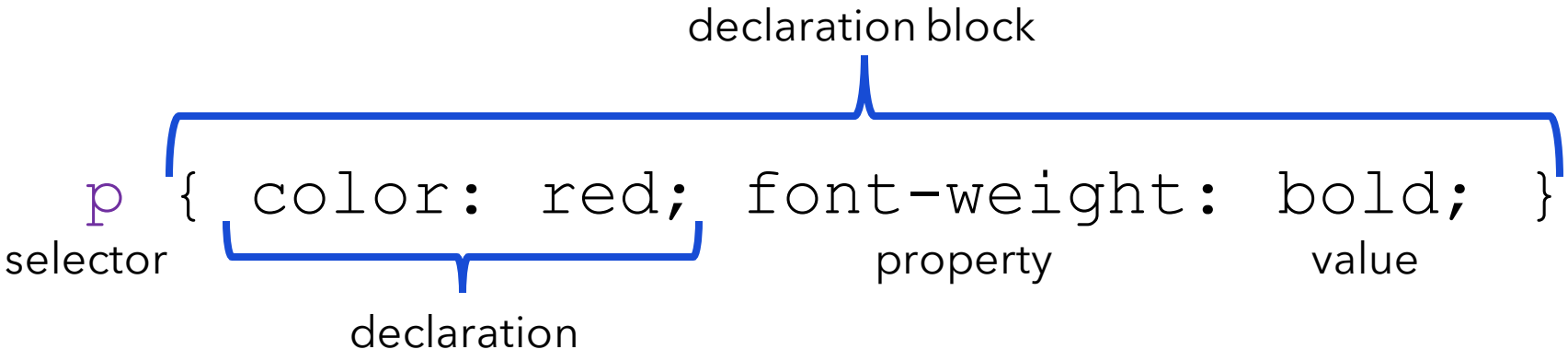
```
<head>
   <link rel="stylesheet"
      type="text/css"
      href="style.css" />
</head>
<body>
   <p>This is a centered
      paragraph of bold red
      text.</p>
</body>
```

style.css

```
p {
     text-align: center;
     color: red;
     font-weight: bold;
}
...
```

CSS Rules - Structure

- CSS rules contain two main parts:
  - **Selector** – a pattern used to select which HTML element(s) to be styled
  - **Declaration** – description of how to style selected elements



CSS Rules - Structure

- A selector can have multiple declarations
- Multiple selectors can share a declaration block

```
h1, h2, h3, p {
    text-align: center;
    color: red;
    font-weight: bold;
}
```

Types of Selectors

- **Element** – refers to an HTML element
  - Also known as simple selector or type selector

```
h1, p, div { color: red; font-weight: bold; }
```

- **ID** – refers to one HTML element on the page with that id attribute
  - Each page can only have one element with a given id (unique)

```
#intro { font-family: Arial, sans serif; }
```

```
<p id="intro">Intro text goes here.</p>
```

Types of Selectors

- **Class** – selects elements with a particular class attribute
  - Can apply to multiple elements of the given type

```
p.warning { color: red; font-weight: bold; }
```

```
<p class="warning">Warning text goes here.</p>
```

- **Generic Class** – selects any element with a particular class attribute
  - Can apply to multiple elements of any type

```
.highlight { font-weight: bold; font-size: 18px }
```

```
<h4 class="highlight">Highlight heading.</h4>
<p class="highlight">Highlight text.</p>
```

Types of Selectors

- **Pseudo-class** – selects elements when they are in a particular state
  - Ex: when something happens in the browser
  - Many pseudo-classes
    - `:active` – elements activated by user (for click, between mouse down and up)
    - `:checked` – radio, checkbox, option elements checked by user
    - `:disabled` – elements that cannot receive focus
    - `:focus` – element that has the user's focus
    - `:hover` – elements currently hovered over by mouse
    - `:link` – link element that has not yet been visited
    - `:visited` – link element that has been visited

```
a:hover { color: red; text-decoration: none; }
```

Types of Selectors

- **Pseudo-class** – selects elements when they are in a particular state
  - Some help us select a pattern of children
    - `:first-child`
    - `:last-child`
    - `:nth-child`
    - `:nth-last-child`
    - `:first-of-type`
    - `:last-of-type`
    - `:nth-of-type`
    - `:nth-last-of-type`

Make even rows of a table shaded light cyan

```
tr:nth-child(even){
    background-color: LightCyan;
}
```

Types of Selectors

- **Pseudo-element** – selects part of an element to apply the style
  - Many pseudo-element selectors
    - `::after` - just after the element (could be used to add content)
    - `::before` - just before the element (could be used to add content)
    - `::first-letter` - first letter of the element text
    - `::first-line` - first line of the element text
    - `::selection` - the selected part of the document
    - `::backdrop` - immediately below the element when rendered in fullscreen

```
h1::first-letter { color: red; font-size: 200%; }
```

Types of Selectors

- **Universal** – selects all elements in the document or all elements inside another element
  - Every element in the HTML

```
* { color: blue; font-size: 12px; }
```

  - Every element inside a `<div>` element

```
div * { color: green; font-size: 11px; }
```

Types of Selectors

- **Attribute** – selects elements based on the presence or value of an attribute
  - `[attribute]` – select all elements with attribute present
  - `[attribute=match]` – select all elements that have attribute with this value (equals)
  - `[attribute^=match]` – select all elements that have attribute beginning with this value
  - `[attribute$=match]` - select all elements that have attribute ending with this value
  - `[attribute*=match]` – select all elements that have attribute containing this value

Types of Selectors

- **Attribute** – selects elements based on the presence or value of an attribute
  - Most useful for form elements (UI for validation and error handling)

```
[value] { background-color: yellow; color: red; }

input[value] { background-color: yellow; color: red; }
```

```
<form action="action.php" method="post">
   <label for="fname">First Name:</label>
   <input type="text" id="fname" name="fname" value="Julia" />
</form>
```

Combining Selectors

Selectors can be combined to refine the document elements that are selected based on HTML document hierarchy

- **Descendant selectors** – space separated selectors
  - Any descendent in the tree will be selected
  - Ex: any `span` inside of a `div` (at any level)

```
div span { color: green; font-size: 11px; }
```

Combining Selectors

- **Direct child selectors** – selectors separated by **>**
  - Only immediate children will be selected
  - Ex: only `p` elements that are direct children of a `div`

```
div > p { color: green; font-size: 11px; }
```

Combining Selectors

- **Adjacent sibling selectors** – selectors separated by **+**
  - Targets adjacent sibling of a specified element based on selector
  - Ex: The next `li` element that is a sibling of the `li` element with class `selected`

```
li.selected + li { background-color: deepskyblue; }
```

```
<ul>
  <li>First</li>
  <li class="selected">Second</li>
  <li>Third</li>
  <li>Fourth</li>
</ul>
```

Combining Selectors

- **General sibling selectors** – selectors separated by ~
  - Targets all elements that are next siblings of a specified element
  - Ex: All next sibling `li` elements of the `li` element with class `selected`

```
li.selected ~ li { background-color: deepskyblue; }
```

```
<ul>
  <li>First</li>
  <li class="selected">Second</li>
  <li>Third</li>
  <li>Fourth</li>
</ul>
```

In Class Activity

CSS Selectors

Cascading Style Sheets

- What if multiple rules apply to a given element?
  - Which one is chosen? How does the element get styled?
- They all get applied, but factors determine which will take precedence!
  - Think: inheritance – in Java, the child classes take precedence

Cascading Style Sheets

- Cascade: the order of CSS rules matter
  - When two rules have equal "specificity," then the later rule will be used
- Specificity: a weight calculated for a given CSS declaration
  - Rules with higher specificity will be chosen

Cascading Style Sheets

- Consider these rules, which would take preference?

```
h1 {
        color: green;
}

h1 {
        color: red;
}
```

Cascade!

```
<h1 class="heading">An Example Heading.</h1>
```

Cascading Style Sheets

- Consider these rules, which would take preference?

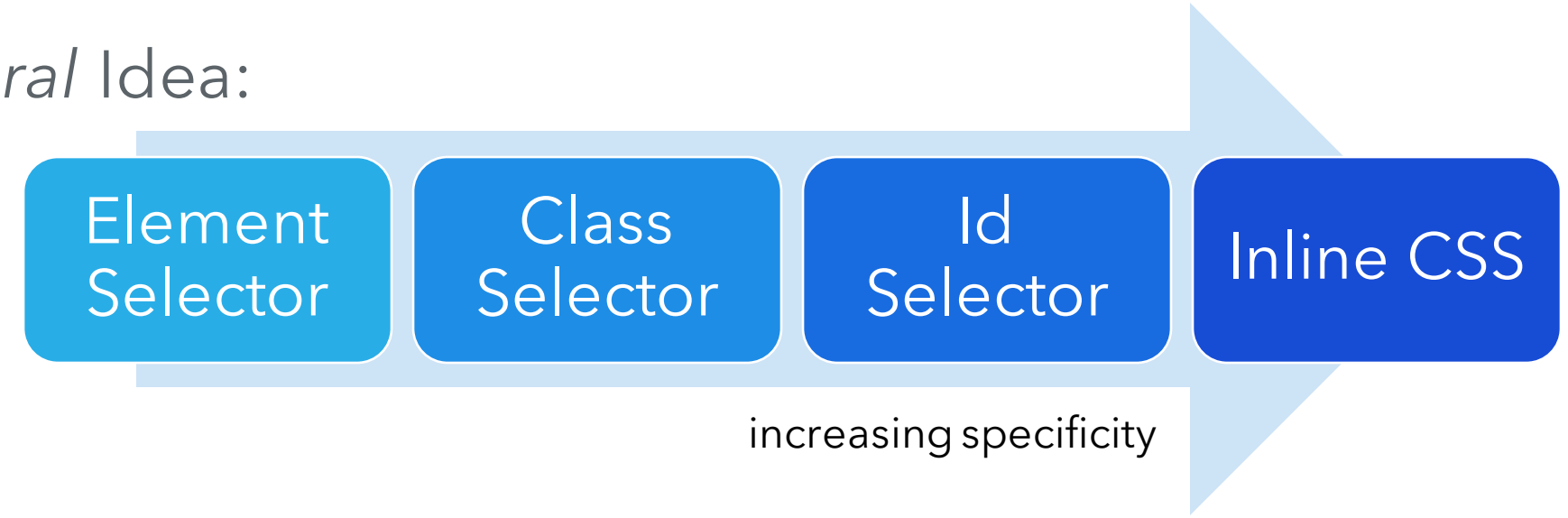
```
h1.heading {
        color: green;
}

h1 {
        color: red;
}
```

Specificity!

```
<h1 class="heading">An Example Heading.</h1>
```

Specificity

- *General* Idea:



  - Roughly: more selectors = has more precedence
- In reality: *Much more nuanced!*

Specificity

inline styles	id selectors	class, attribute, pseudo-class selectors	element, pseudo-element selectors
Defined in the html `style=""` attribute	Number of each in the overall selector	Number of each in the overall selector	Number of each in the overall selector

Specificity

```
<h1 style="font-family: Arial">An Example Heading.</h1>
```

Specificity

```
h1.heading { color: green; }
```

Specificity

```
div > span { font-weight: bold; }
```

Specificity

```
#outer div ul li a { text-decoration: none; }
```

In Class Activity

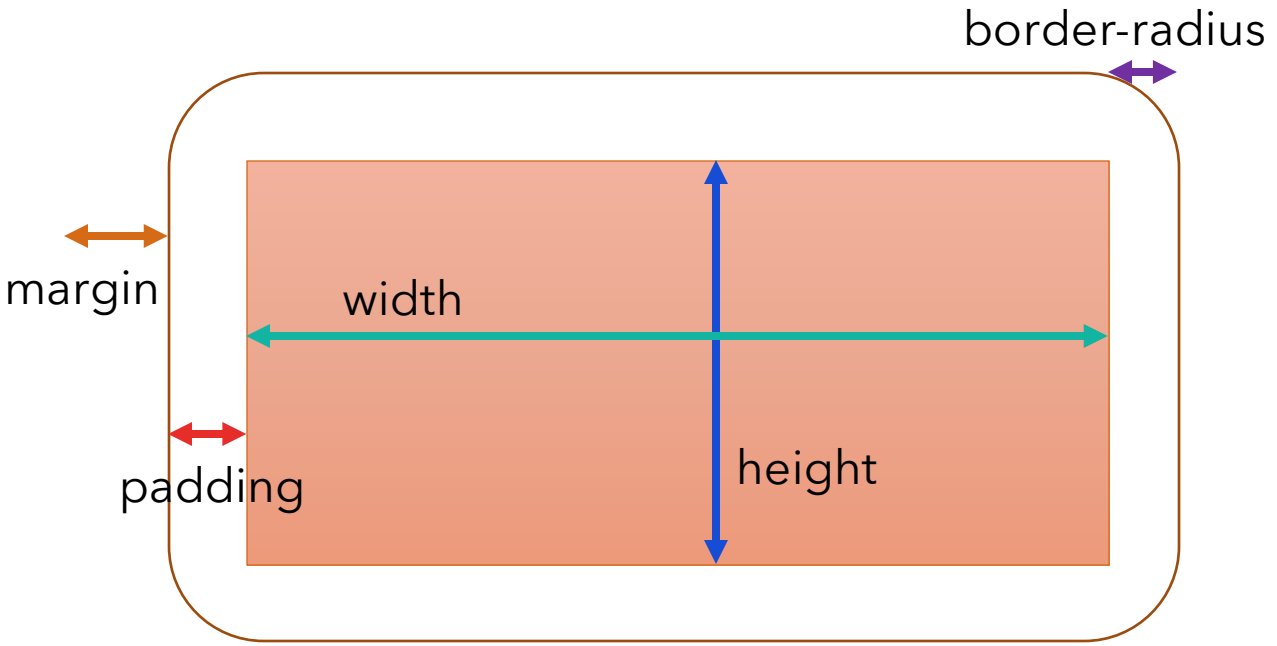
Determining Specificity

CSS Common Properties

https://cs4640.cs.virginia.edu/resources/css-rules/

CSS – Box Properties

- Boxes by default are just large enough to fit contents
- Padding adds space inside box
- Margin sets spacing outside box
- Specify size in pixels (`px`) or percent (`%`)
  - Percent is relative to container's dimensions



Responsive Design in CSS

Flexbox

- CSS3 Web Layout Model
  - W3C Candidate Recommendation
  - https://www.w3.org/TR/css-flexbox-1/
- One-Dimensional layout – rows or columns
  - Items flex / expand or shrink to fit space
  - Supports wrapping based on screen size (responsive)

https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox

Flexbox

- Set display to flex on container element

```
#container { display: flex; }
```

- Two directions for display (row vs column)

```
#container { flex-direction: row; }
```

https://mdn.github.io/learning-area/css/css-layout/flexbox/flexbox0.html
https://mdn.github.io/learning-area/css/css-layout/flexbox/flexbox-wrap0.html

Flexbox

- Set the inner elements to determine their flex width or basis

```
.flexitem { flex: 200px; }
```

```
.flexitem { flex-basis: 25%; }
```

- Set wrapping on container to allow flex items to flow

```
#container { flex-wrap: wrap; }
```

- Browser will determine best way to flow based on screen size

https://mdn.github.io/learning-area/css/css-layout/flexbox/flexbox0.html
https://mdn.github.io/learning-area/css/css-layout/flexbox/flexbox-wrap0.html

Flexbox Demo

Flexbox

- Provides flexibility and responsive design
- Somewhat limited on overall layout

We have other options!

Gridding System

- Consider the page as a grid: rows and columns
  - We will determine how many rows we need for our design
  - Divide the page into 12 columns
    - Each row will have 12 equal-width columns
- Encode these in CSS rules
  - Uses classes to denote columns and column spans
    - .col-4 is a block element that spans 4 columns
    - .col-12 is a block element that spans 12 columns
    - .row is a block element that spans 12 columns

Demo

Gridding System

- Use `@media` queries to provide layout instructions to browser based on screen size (and type)

- only
- not

- all – any type
- screen
- print
- speech

- max-width
- min-width
- defined in pixels
- check responsive design mode

```
@media only screen and (max-width: 768px) {
      .col-4 { width: 100%; }
}
```

Demo

Gridding System

- This is the kind of layout that Bootstrap provides!
  - Next time