HTML & CSS

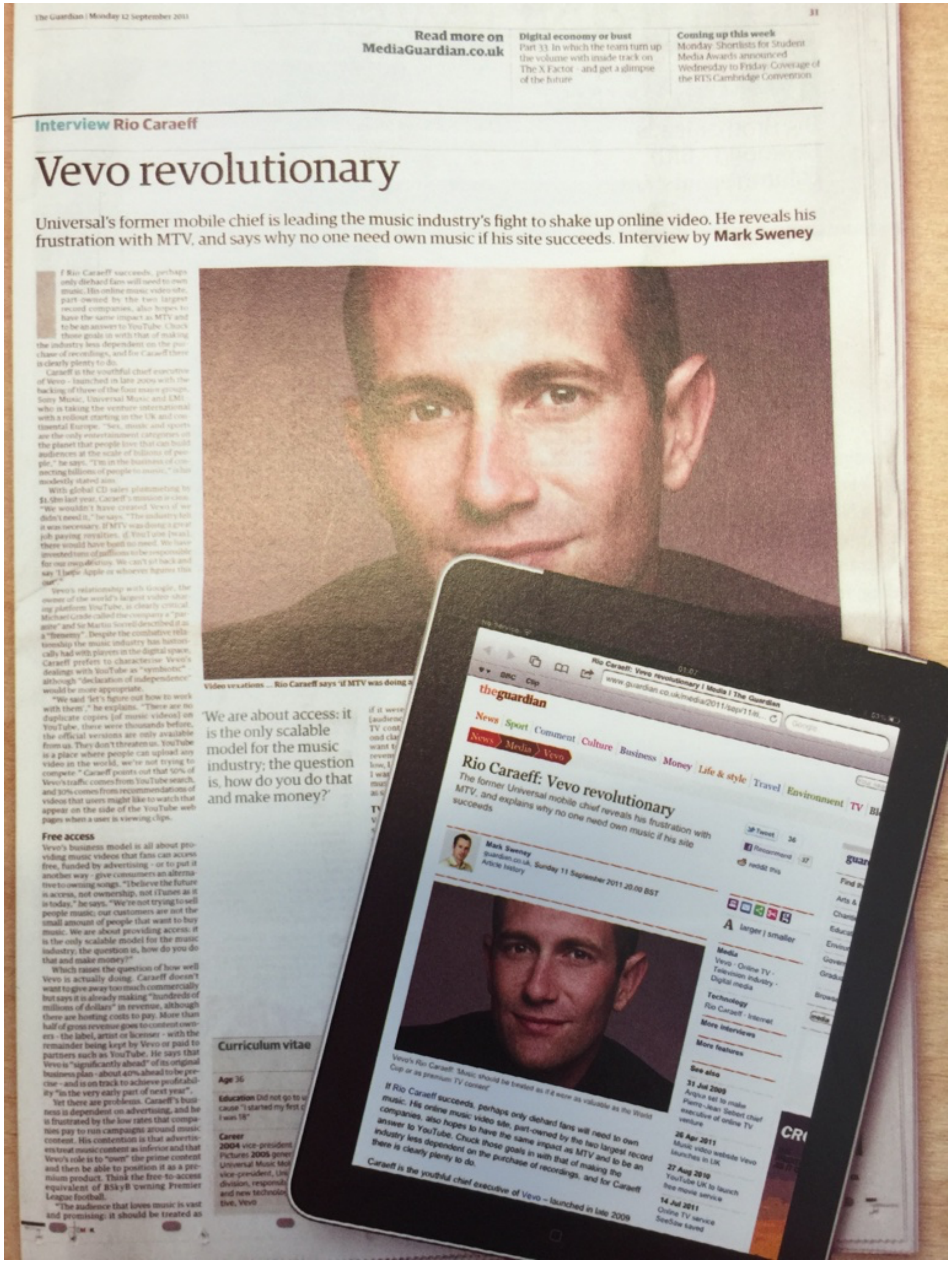
SWE 432, Fall 2016

Design and Implementation of Software for the Web

GEORGE MASON UNIVERSITY

HTML: HyperText Markup Language

- Language for describing *structure* of a document
- Denotes hierarchy of elements
- What might be elements in this document?



The Guardian | Monday 12 September 2011

Read more on MediaGuardian.co.uk

Digital economy or bust Part 33. In which the team turn up the volume with inside track on The X Factor - and get a glimpse of the future

Coming up this week Monday: Shortlists for Student Media Awards announced Wednesday to Friday: Coverage of the RTS Cambridge Convention

Interview Rio Caraeff

Vevo revolutionary

Universal's former mobile chief is leading the music industry's fight to shake up online video. He reveals his frustration with MTV, and says why no one need own music if his site succeeds. Interview by **Mark Sweney**

Video vexations ... Rio Caraeff says 'if MTV was doing a

'We are about access: it is the only scalable model for the music industry; the question is, how do you do that and make money?'

Free access

Curriculum vitae

theguardian

News Sport Comment Culture Business Money Life & style Travel Environment TV

News › Media › Vevo

Rio Caraeff: Vevo revolutionary

The former Universal mobile chief reveals his frustration with MTV, and explains why no one need own music if his site succeeds

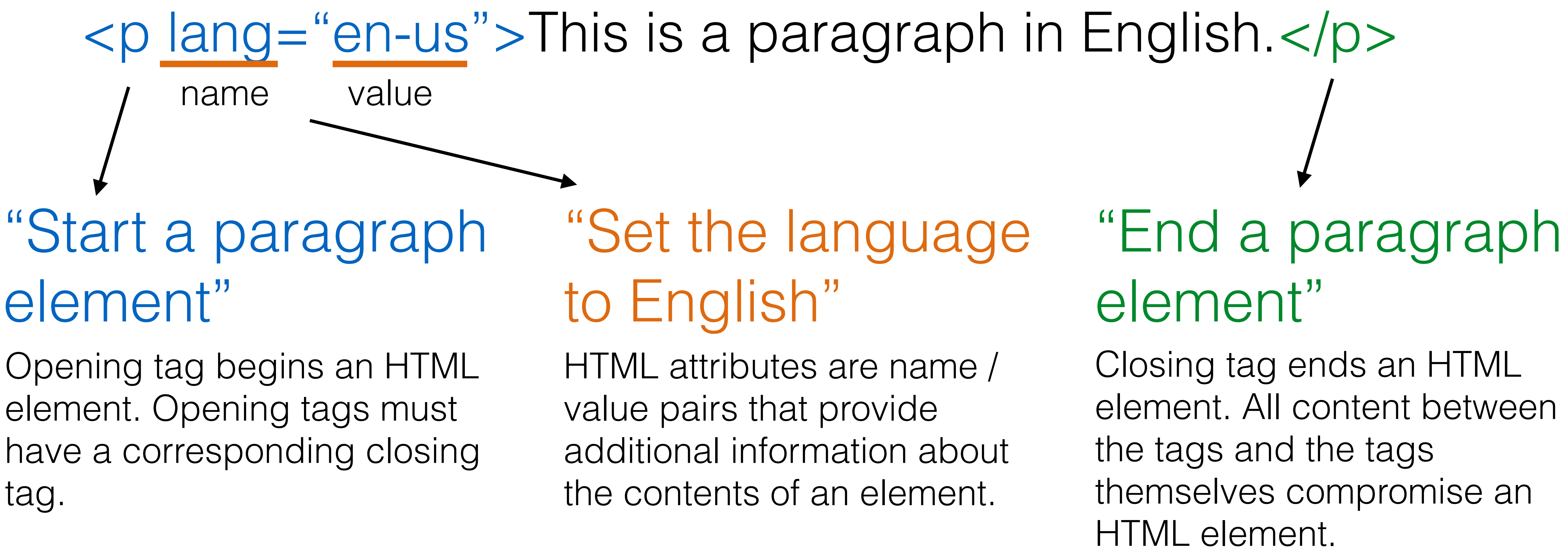
Mark Sweney
guardian.co.uk, Sunday 11 September 2011 20.00 BST
Article history

www.guardian.co.uk/media/2011/sep/11/...
Rio Caraeff: Vevo revolutionary | Media | The Guardian
Tweet
Recommend
reddit this
A larger | smaller
Media
Vevo · Online TV · Television industry · Digital media

HTML History

- 1995: HTML 2.0. Published as standard with RFC 1866
- 1997: HTML 4.0 Standardized most modern HTML element w/ W3C recommendation
  - Encouraged use of CSS for styling elements over HTML attributes
- 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
    - e.g., New tags that explain *purpose* of content
  - Important changes to DOM (will see these later….)

HTML Elements



<p lang=“en-us”>This is a paragraph in English.</p>

name

value

“Start a paragraph element”

Opening tag begins an HTML element. Opening tags must have a corresponding closing tag.

“Set the language to English”

HTML attributes are name / value pairs that provide additional information about the contents of an element.

“End a paragraph element”

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

- Open tag, close tag

HTML Elements

<input type=“text” />

“Begin and end input element”

Some HTML tags can be self closing, including a built-in closing tag.

<!-- This is a comment. Comments can be multiline. -->

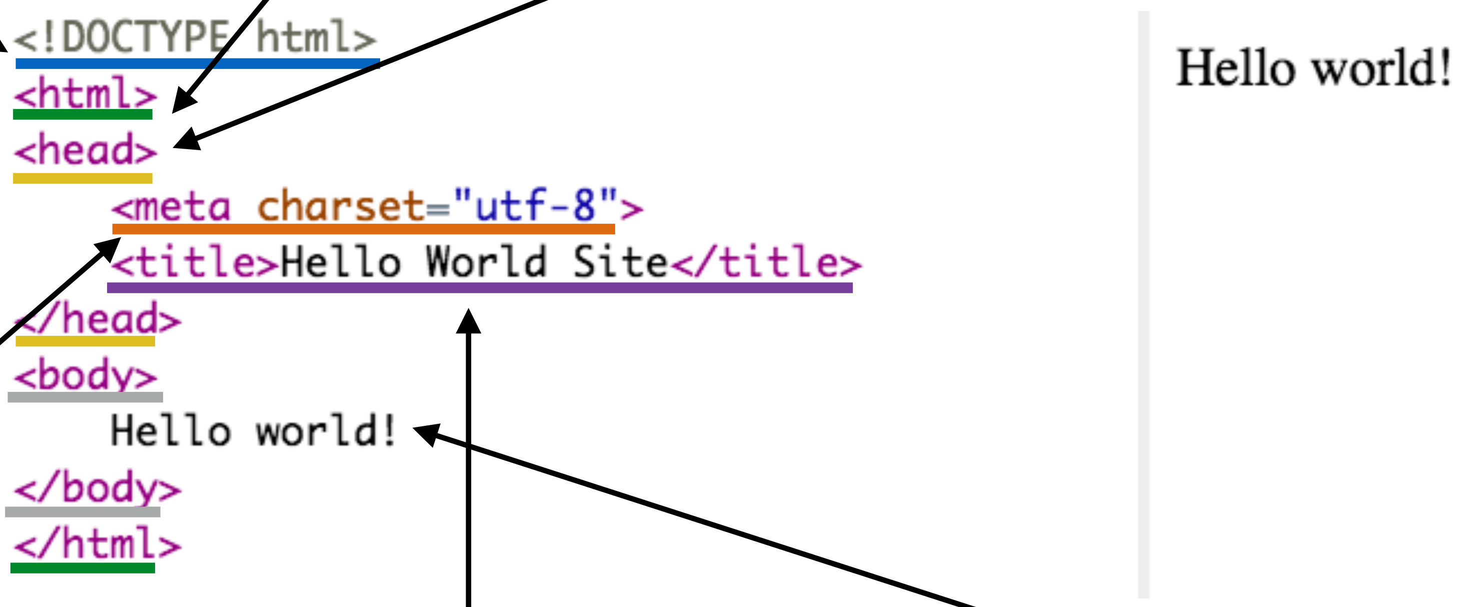
A starter HTML document

“Use HTML5 standards mode”

“HTML content”

“Header”

Information *about* the page



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Hello World Site</title>
</head>
<body>
    Hello world!
</body>
</html>
```

Hello world!

“Interpret bytes as UTF-8 characters”

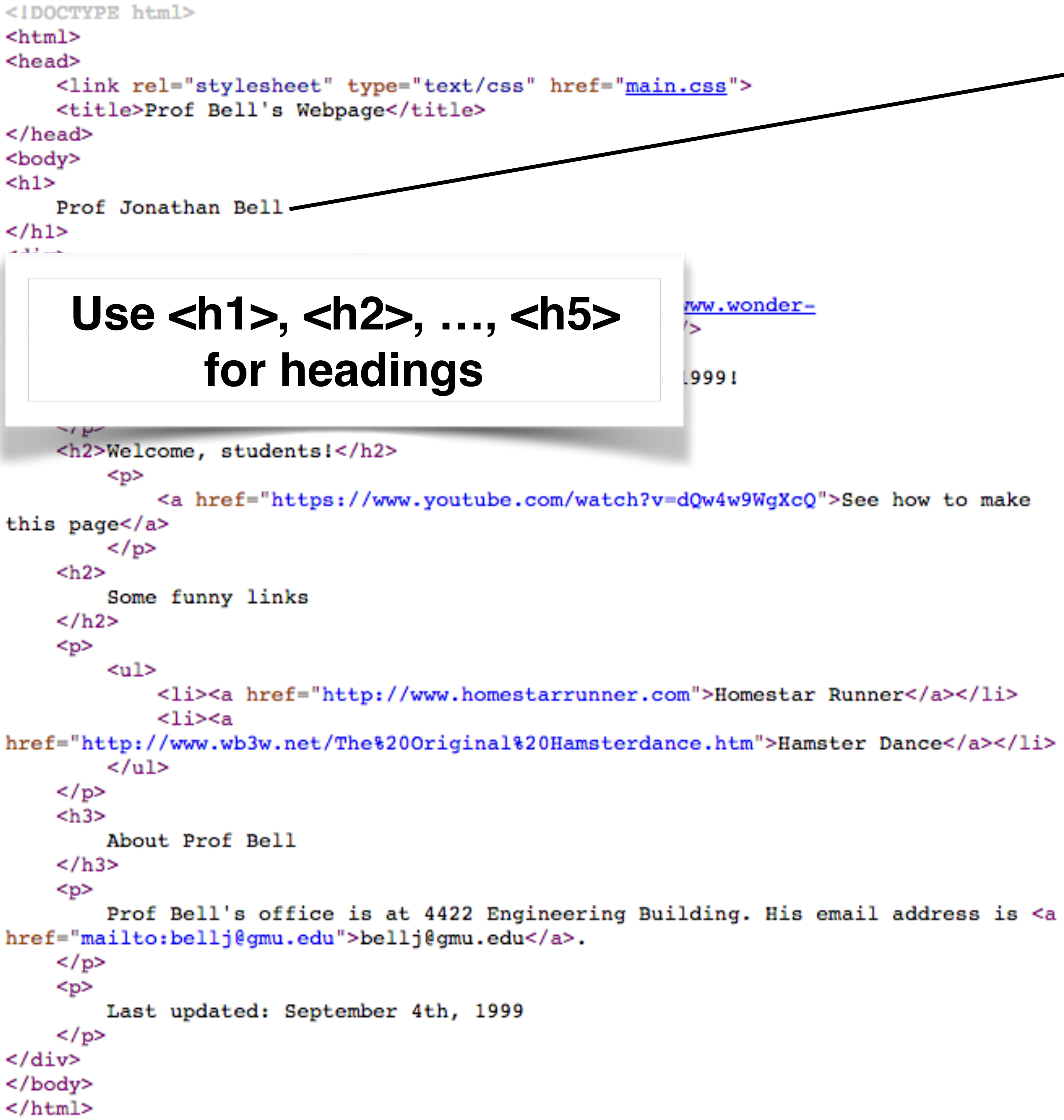
Includes both ASCII & international characters.

“Title”

Used by browser for title bar or tab.

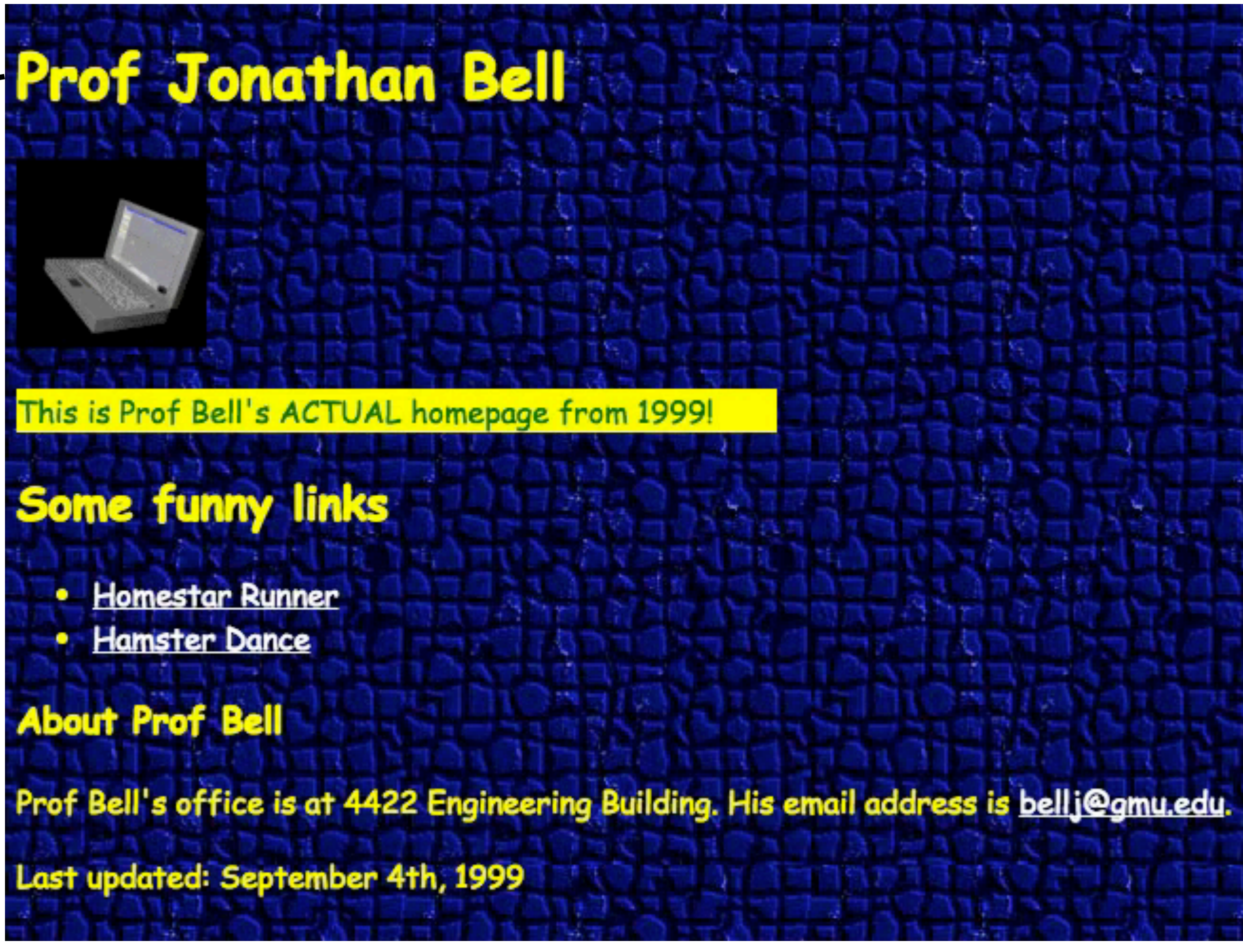
“Document content”

HTML Example



```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="main.css">
    <title>Prof Bell's Webpage</title>
</head>
<body>
<h1>
    Prof Jonathan Bell
</h1>
<div>
                                                       ww.wonder-
                                                       />

                                                       1999!

    </p>
    <h2>Welcome, students!</h2>
        <p>
            <a href="https://www.youtube.com/watch?v=dQw4w9WgXcQ">See how to make
this page</a>
        </p>
    <h2>
        Some funny links
    </h2>
    <p>
        <ul>
            <li><a href="http://www.homestarrunner.com">Homestar Runner</a></li>
            <li><a
href="http://www.wb3w.net/The%20Original%20Hamsterdance.htm">Hamster Dance</a></li>
        </ul>
    </p>
    <h3>
        About Prof Bell
    </h3>
    <p>
        Prof Bell's office is at 4422 Engineering Building. His email address is <a
href="mailto:bellj@gmu.edu">bellj@gmu.edu</a>.
    </p>
    <p>
        Last updated: September 4th, 1999
    </p>
</div>
</body>
</html>
```

Use <h1>, <h2>, …, <h5> for headings



Prof Jonathan Bell

This is Prof Bell's ACTUAL homepage from 1999!

Some funny links

- Homestar Runner
- Hamster Dance

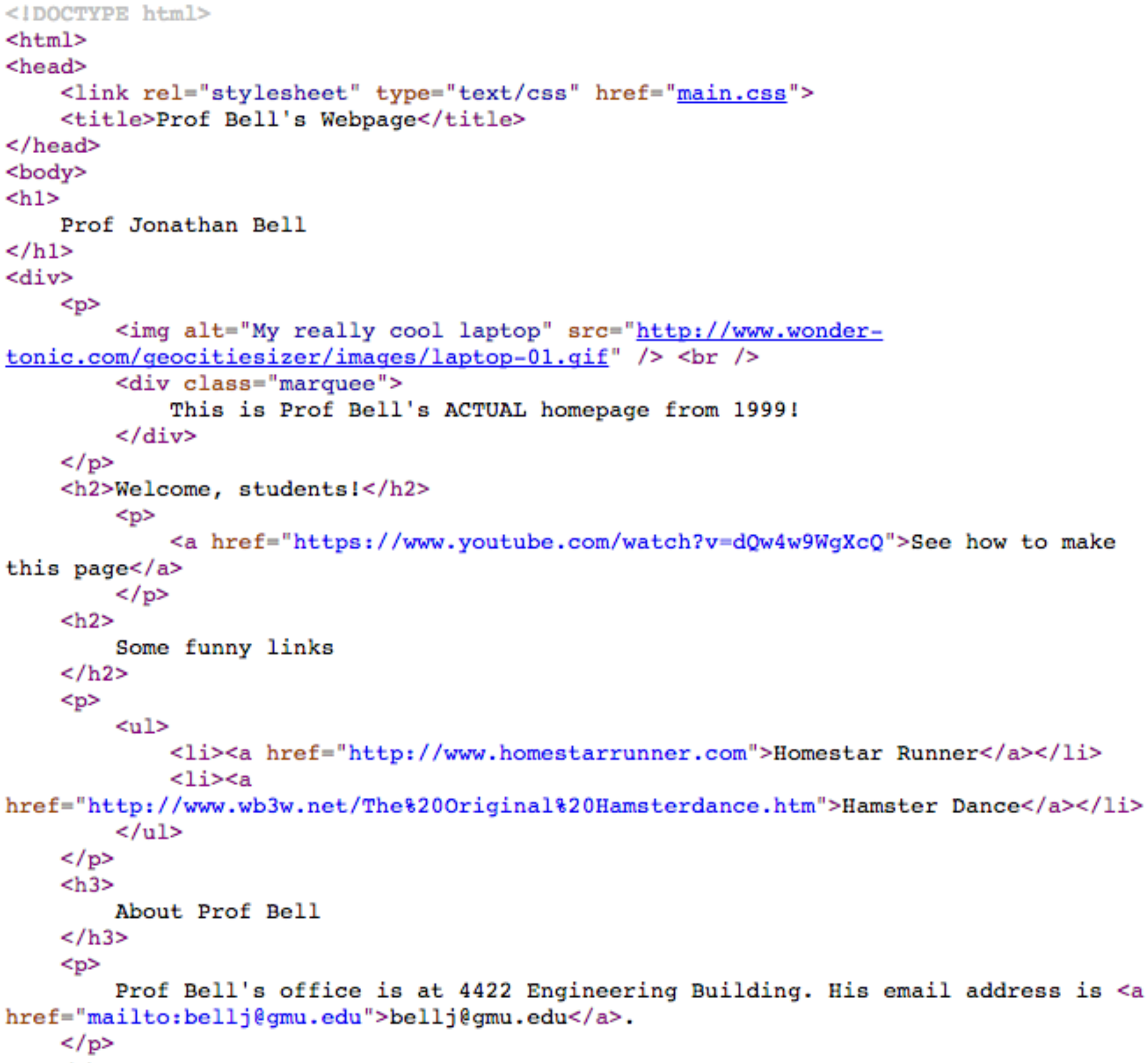
About Prof Bell

Prof Bell's office is at 4422 Engineering Building. His email address is bellj@gmu.edu.

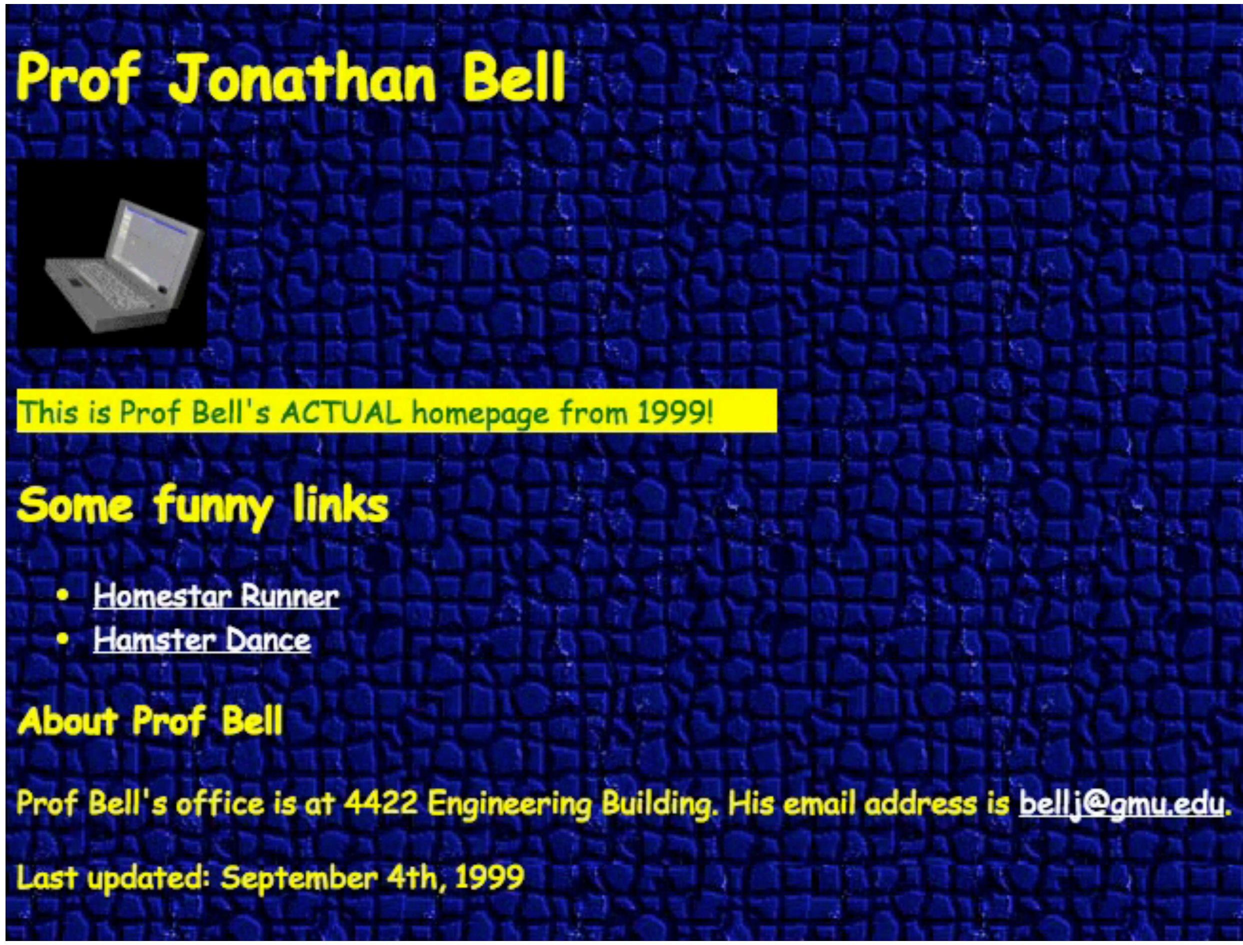
Last updated: September 4th, 1999

https://seecode.run/#-KQgR7vG9Ds7IUJS1kdq

HTML Example



```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="main.css">
    <title>Prof Bell's Webpage</title>
</head>
<body>
<h1>
    Prof Jonathan Bell
</h1>
<div>
    <p>
        <img alt="My really cool laptop" src="http://www.wonder-
tonic.com/geocitiesizer/images/laptop-01.gif" /> <br />
        <div class="marquee">
            This is Prof Bell's ACTUAL homepage from 1999!
        </div>
    </p>
    <h2>Welcome, students!</h2>
        <p>
            <a href="https://www.youtube.com/watch?v=dQw4w9WgXcQ">See how to make
this page</a>
        </p>
    <h2>
        Some funny links
    </h2>
    <p>
        <ul>
            <li><a href="http://www.homestarrunner.com">Homestar Runner</a></li>
            <li><a
href="http://www.wb3w.net/The%20Original%20Hamsterdance.htm">Hamster Dance</a></li>
        </ul>
    </p>
    <h3>
        About Prof Bell
    </h3>
    <p>
        Prof Bell's office is at 4422 Engineering Building. His email address is <a
href="mailto:bellj@gmu.edu">bellj@gmu.edu</a>.
    </p>
```



Prof Jonathan Bell

This is Prof Bell's ACTUAL homepage from 1999!

Some funny links

- Homestar Runner
- Hamster Dance

About Prof Bell

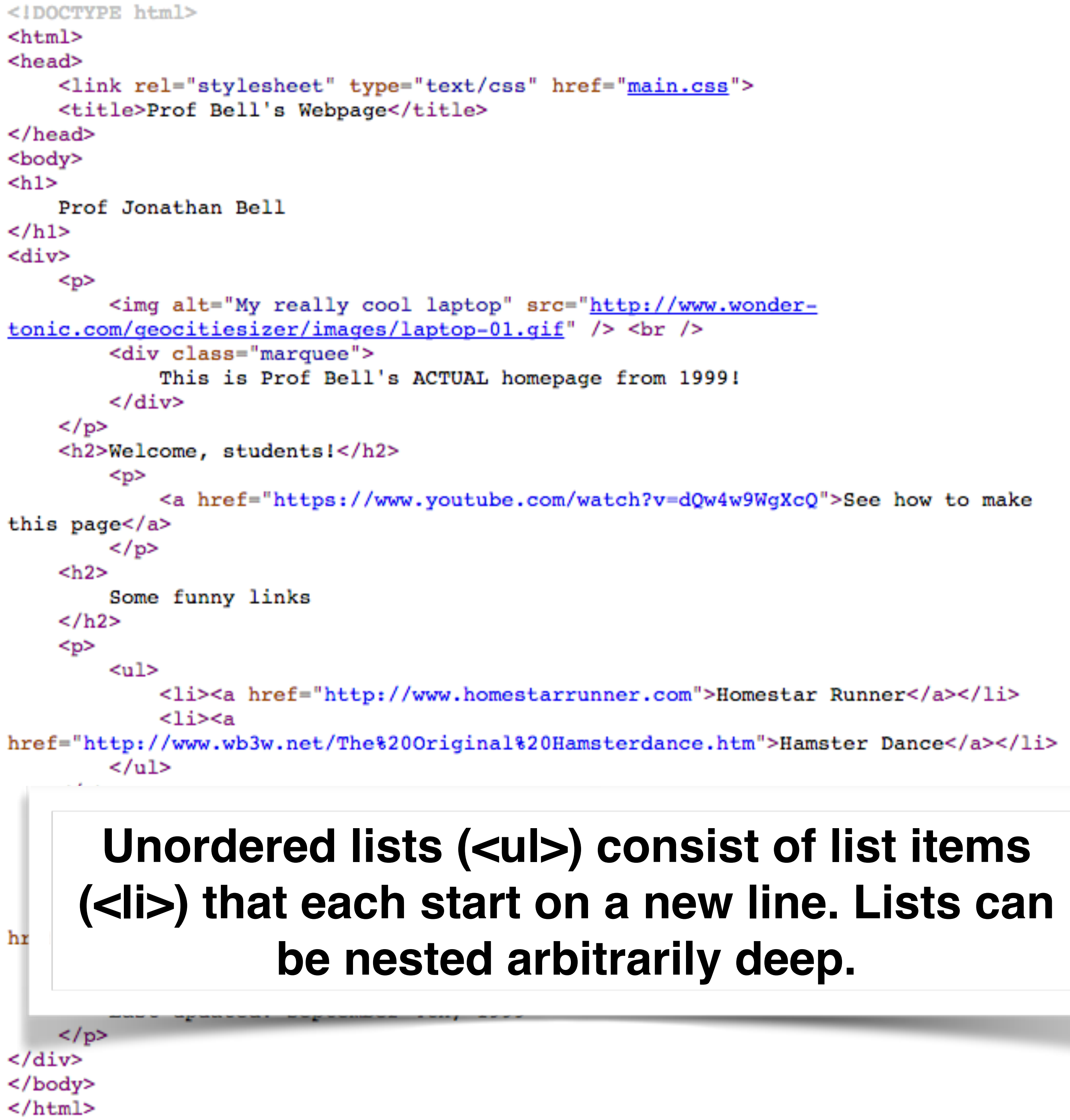
Prof Bell's office is at 4422 Engineering Building. His email address is bellj@gmu.edu.

Last updated: September 4th, 1999

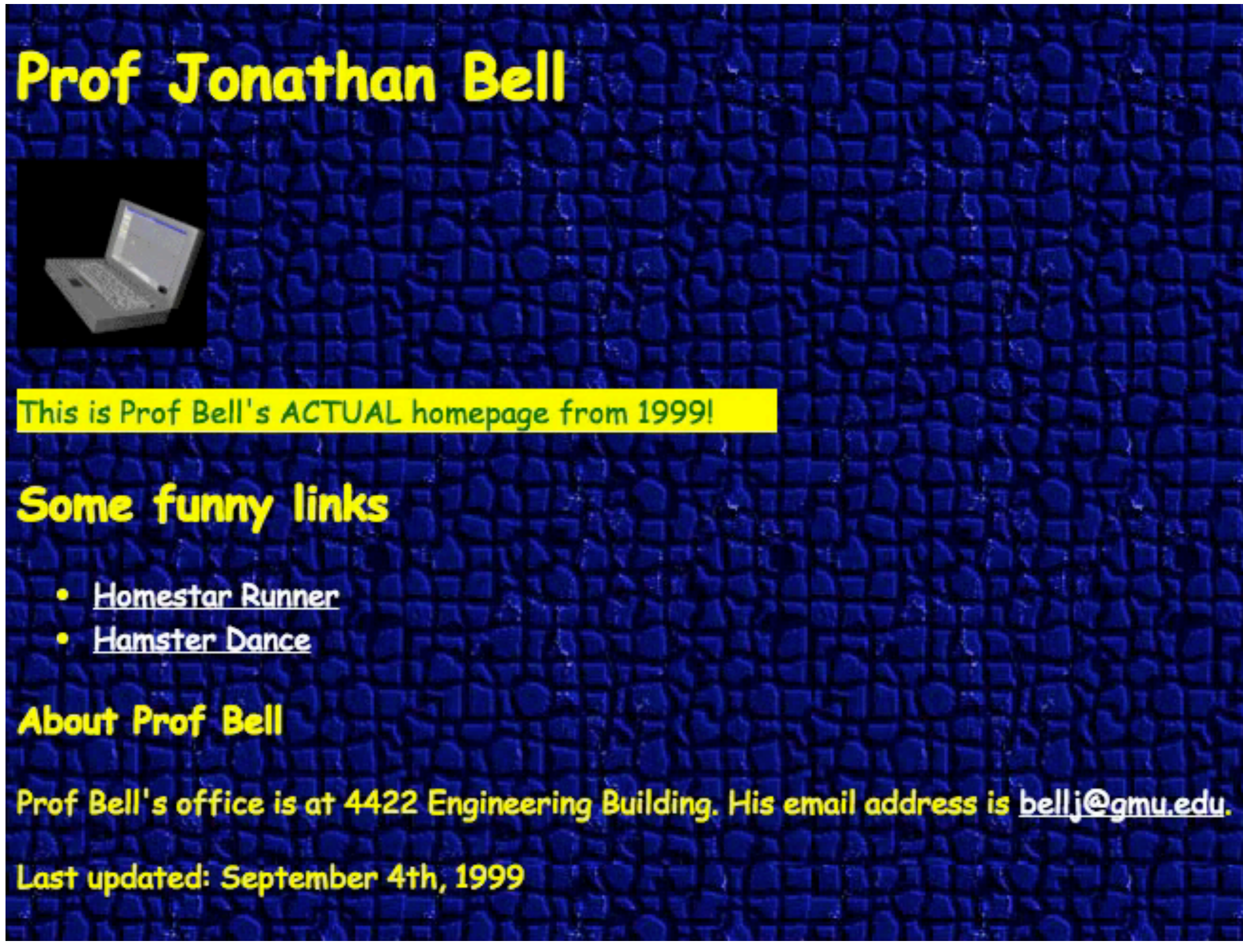
Paragraphs (<p>) consist of related content. By default, each paragraph starts on a new line.

n/#-KQgR7vG9Ds7IUJS1kdq

HTML Example



```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="main.css">
    <title>Prof Bell's Webpage</title>
</head>
<body>
<h1>
    Prof Jonathan Bell
</h1>
<div>
    <p>
        <img alt="My really cool laptop" src="http://www.wonder-
tonic.com/geocitiesizer/images/laptop-01.gif" /> <br />
        <div class="marquee">
            This is Prof Bell's ACTUAL homepage from 1999!
        </div>
    </p>
    <h2>Welcome, students!</h2>
        <p>
            <a href="https://www.youtube.com/watch?v=dQw4w9WgXcQ">See how to make
this page</a>
        </p>
    <h2>
        Some funny links
    </h2>
    <p>
        <ul>
            <li><a href="http://www.homestarrunner.com">Homestar Runner</a></li>
            <li><a
href="http://www.wb3w.net/The%20Original%20Hamsterdance.htm">Hamster Dance</a></li>
        </ul>
hr
    </p>
</div>
</body>
</html>
```



Prof Jonathan Bell

This is Prof Bell's ACTUAL homepage from 1999!

Some funny links

- Homestar Runner
- Hamster Dance

About Prof Bell

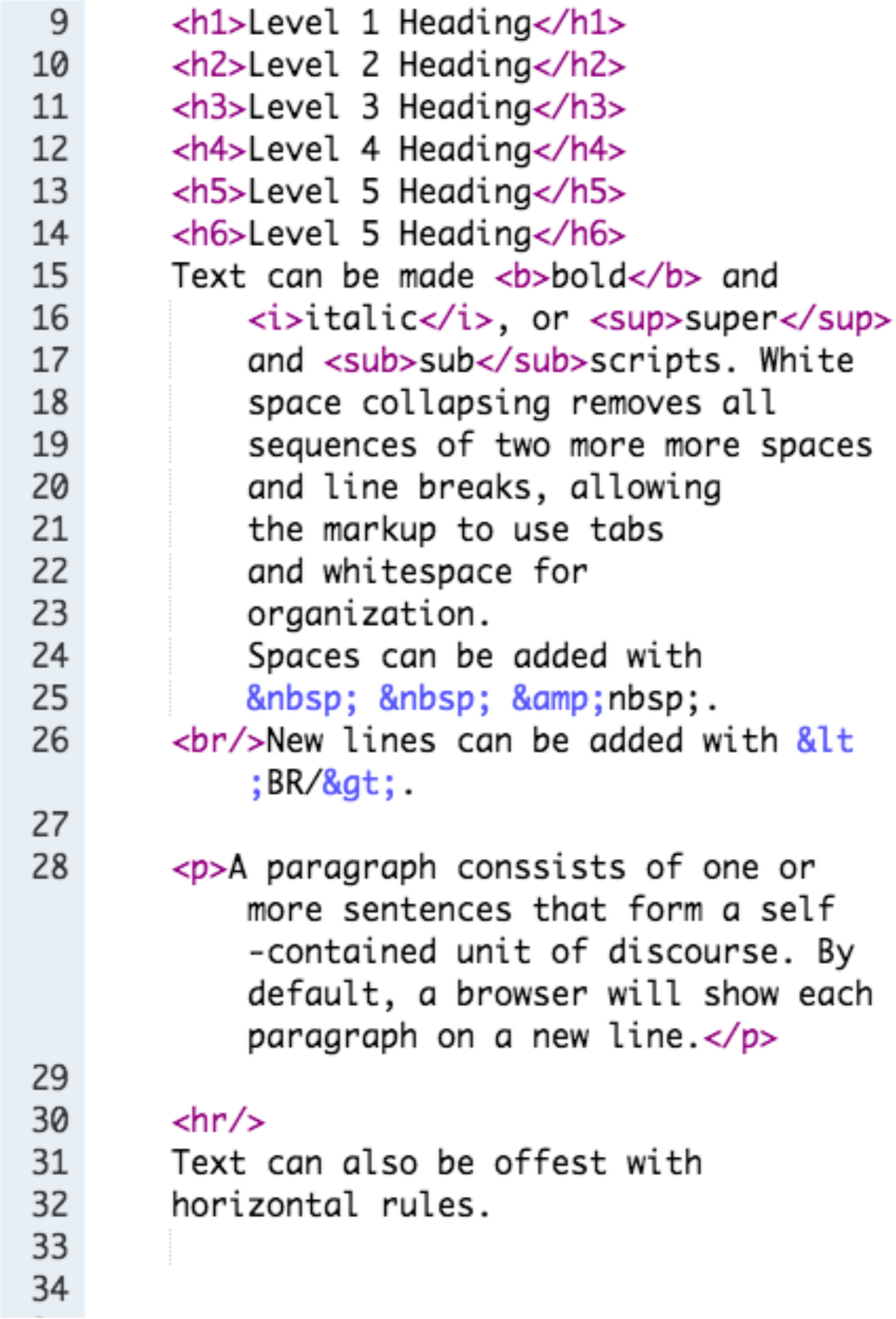
Prof Bell's office is at 4422 Engineering Building. His email address is bellj@gmu.edu.

Last updated: September 4th, 1999

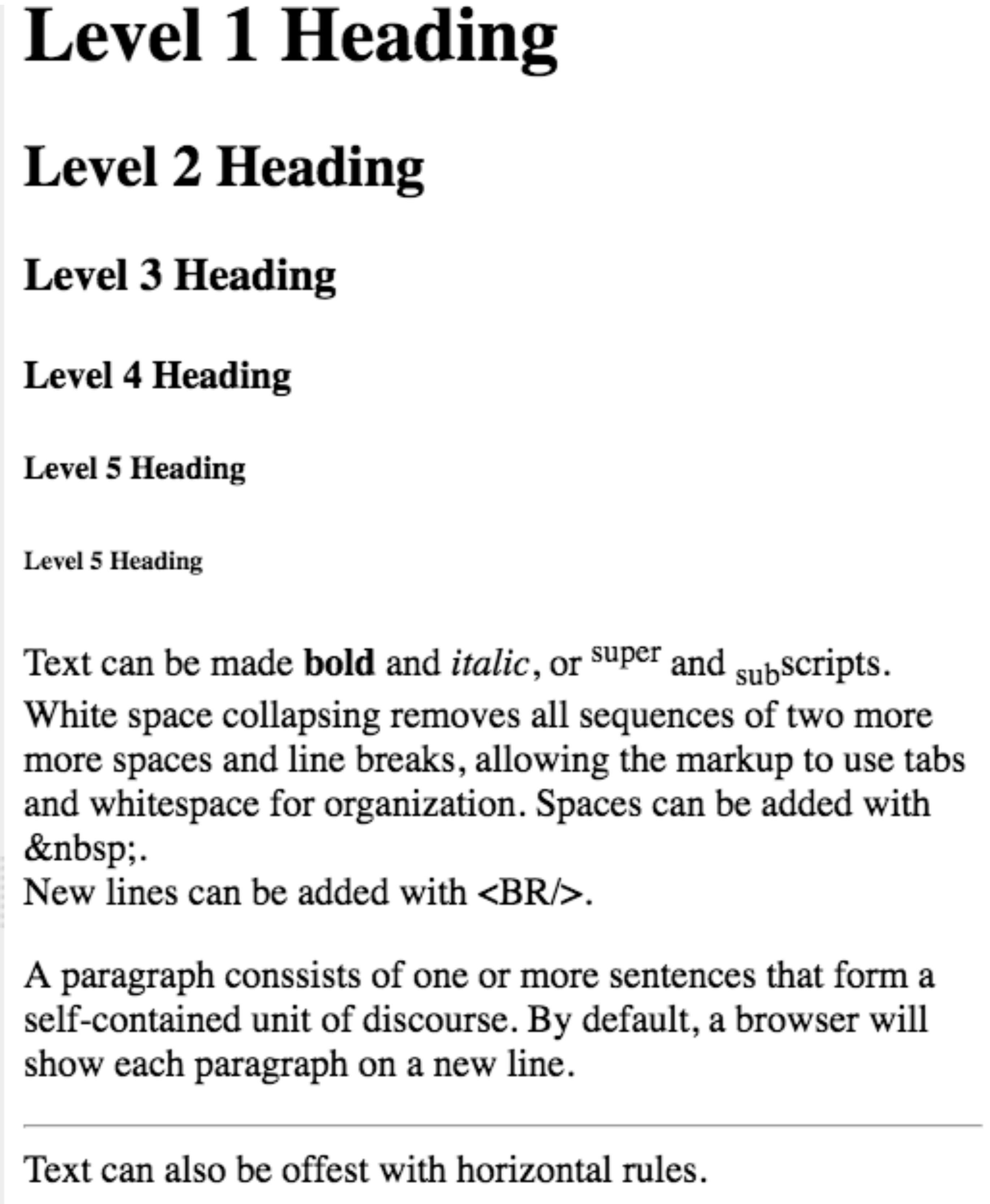
Unordered lists (<ul>) consist of list items (<li>) that each start on a new line. Lists can be nested arbitrarily deep.

https://seecode.run/#-KQgR7vG9Ds7IUJS1kdq

Text



```
<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 5 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 more spaces
    and line breaks, allowing
    the markup to use tabs
    and whitespace for
    organization.
    Spaces can be added with
    &nbsp; &nbsp; &amp;nbsp;.
<br/>New lines can be added with &lt
    ;BR/&gt;.

<p>A paragraph conssists 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 offest with
horizontal rules.

```



Level 1 Heading

Level 2 Heading

Level 3 Heading

Level 4 Heading

Level 5 Heading

Level 5 Heading

Text can be made **bold** and *italic*, or $^{\text{super}}$ and $_{\text{sub}}$scripts. White space collapsing removes all sequences of two more more spaces and line breaks, allowing the markup to use tabs and whitespace for organization. Spaces can be added with .
New lines can be added with <BR/>.

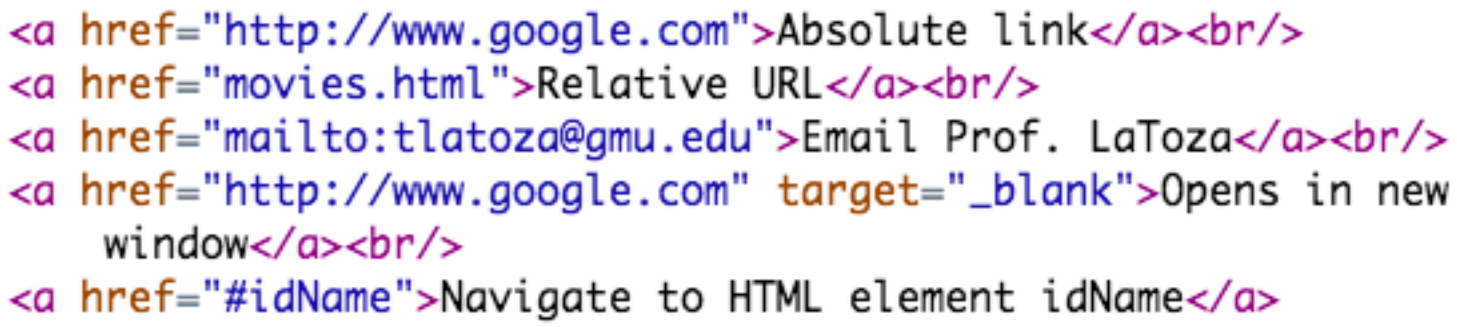
A paragraph conssists of one or more sentences that form a self-contained unit of discourse. By default, a browser will show each paragraph on a new line.

Text can also be offest with horizontal rules.

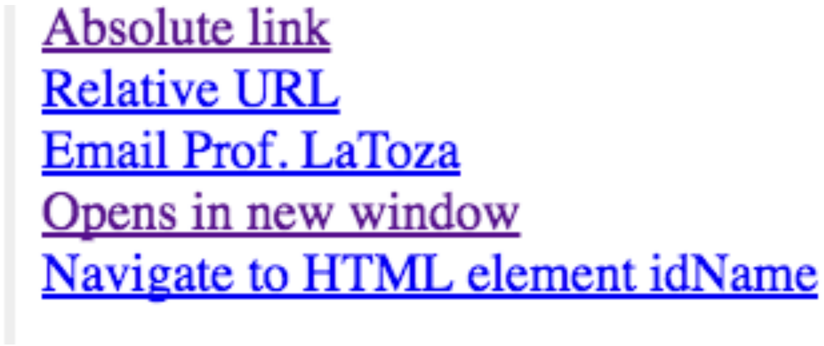
Semantic markup

- Tags that can be used to denote the *meaning* of specific content
- Examples
  - <strong> An element that has importance.
  - <blockquote> An element that is a longer quote.
  - <q> A shorter quote inline in paragraph.
  - <abbr> Abbreviation
  - <cite> Reference to a work.
  - <dfn> The definition of a term.
  - <address> Contact information.
  - <ins><del> Content that was inserted or deleted.
  - <s> Something that is no longer accurate.

Links



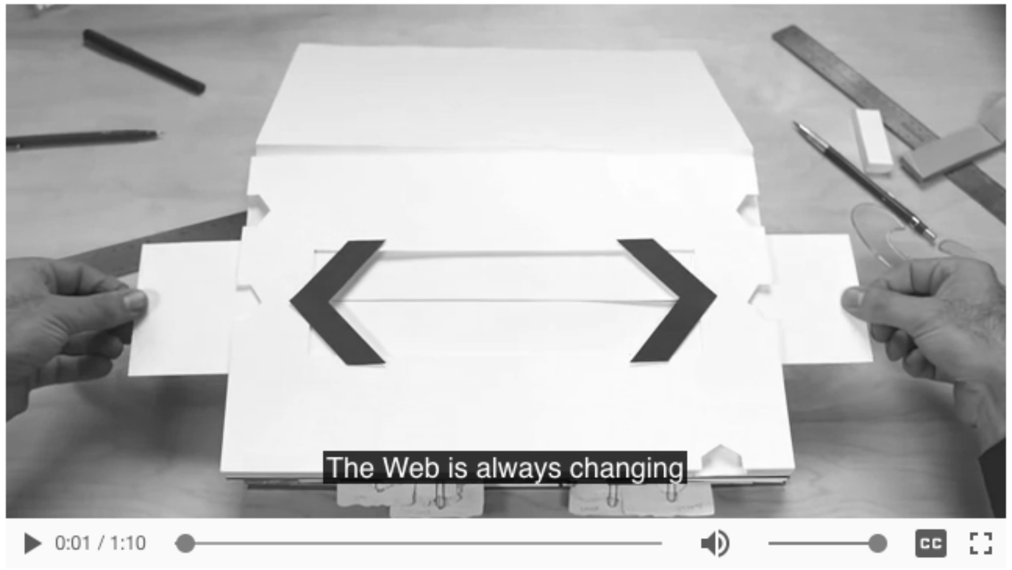
```
<a href="http://www.google.com">Absolute link</a><br/>
<a href="movies.html">Relative URL</a><br/>
<a href="mailto:tlatoza@gmu.edu">Email Prof. LaToza</a><br/>
<a href="http://www.google.com" target="_blank">Opens in new
    window</a><br/>
<a href="#idName">Navigate to HTML element idName</a>
```

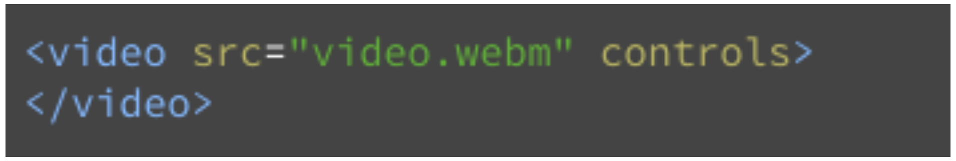


Absolute link
Relative URL
Email Prof. LaToza
Opens in new window
Navigate to HTML element idName

Images, Audio, Video

- HTML includes standard support for <img>, <audio>, <video>
- Common file formats
  - Images: .png, .gif, .jpg
  - Audio: .mp3
  - Video: .mp4

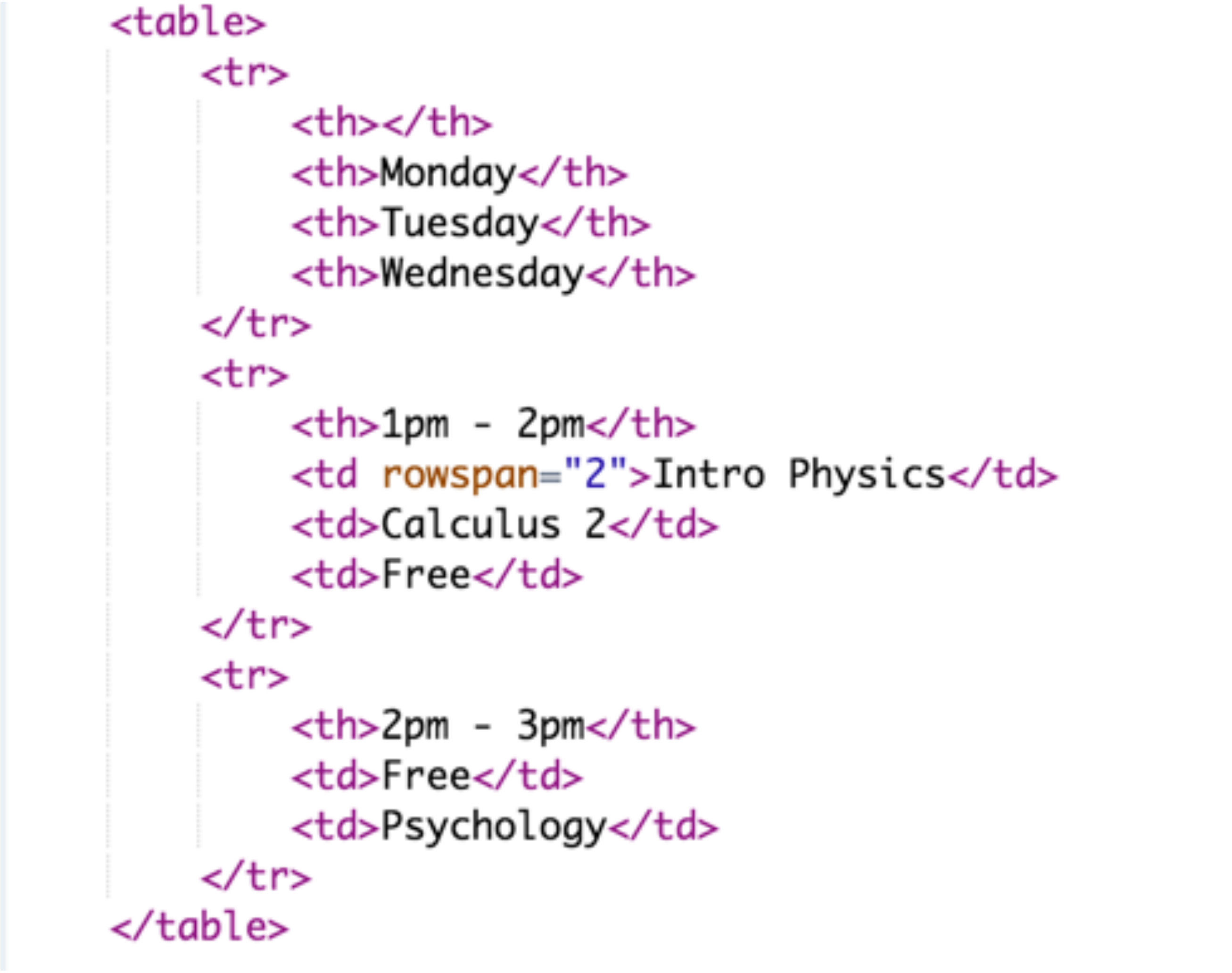




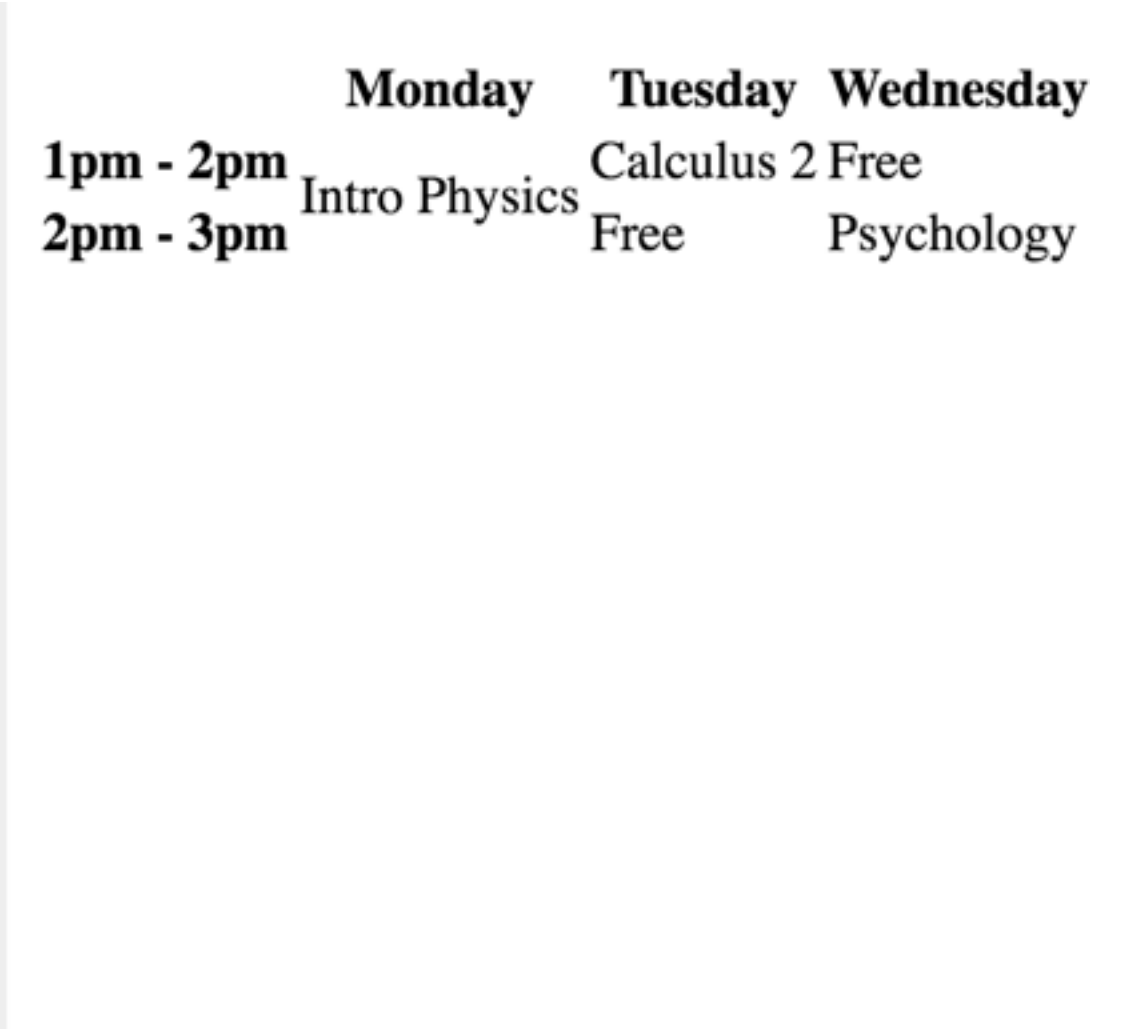
```
<video src="video.webm" controls>
</video>
```

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

Tables

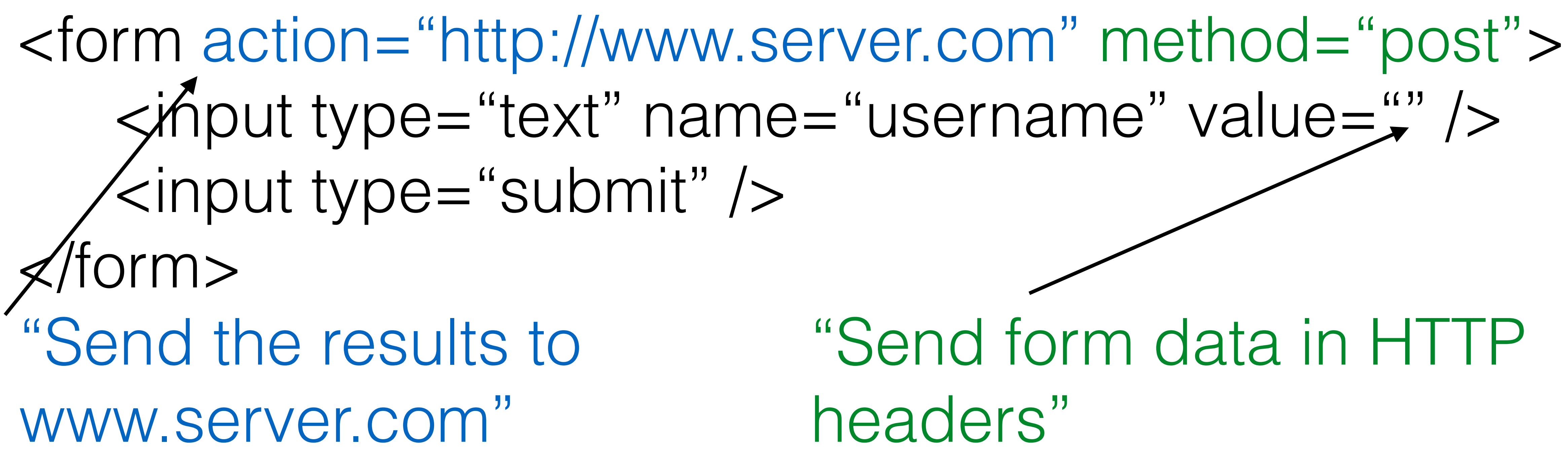


```
<table>
    <tr>
        <th></th>
        <th>Monday</th>
        <th>Tuesday</th>
        <th>Wednesday</th>
    </tr>
    <tr>
        <th>1pm - 2pm</th>
        <td rowspan="2">Intro Physics</td>
        <td>Calculus 2</td>
        <td>Free</td>
    </tr>
    <tr>
        <th>2pm - 3pm</th>
        <td>Free</td>
        <td>Psychology</td>
    </tr>
</table>
```



	Monday	Tuesday	Wednesday
1pm - 2pm	Intro Physics	Calculus 2	Free
2pm - 3pm		Free	Psychology

Forms



```
<form action="http://www.server.com" method="post">
    <input type="text" name="username" value="" />
    <input type="submit" />
</form>
```

"Send the results to www.server.com"

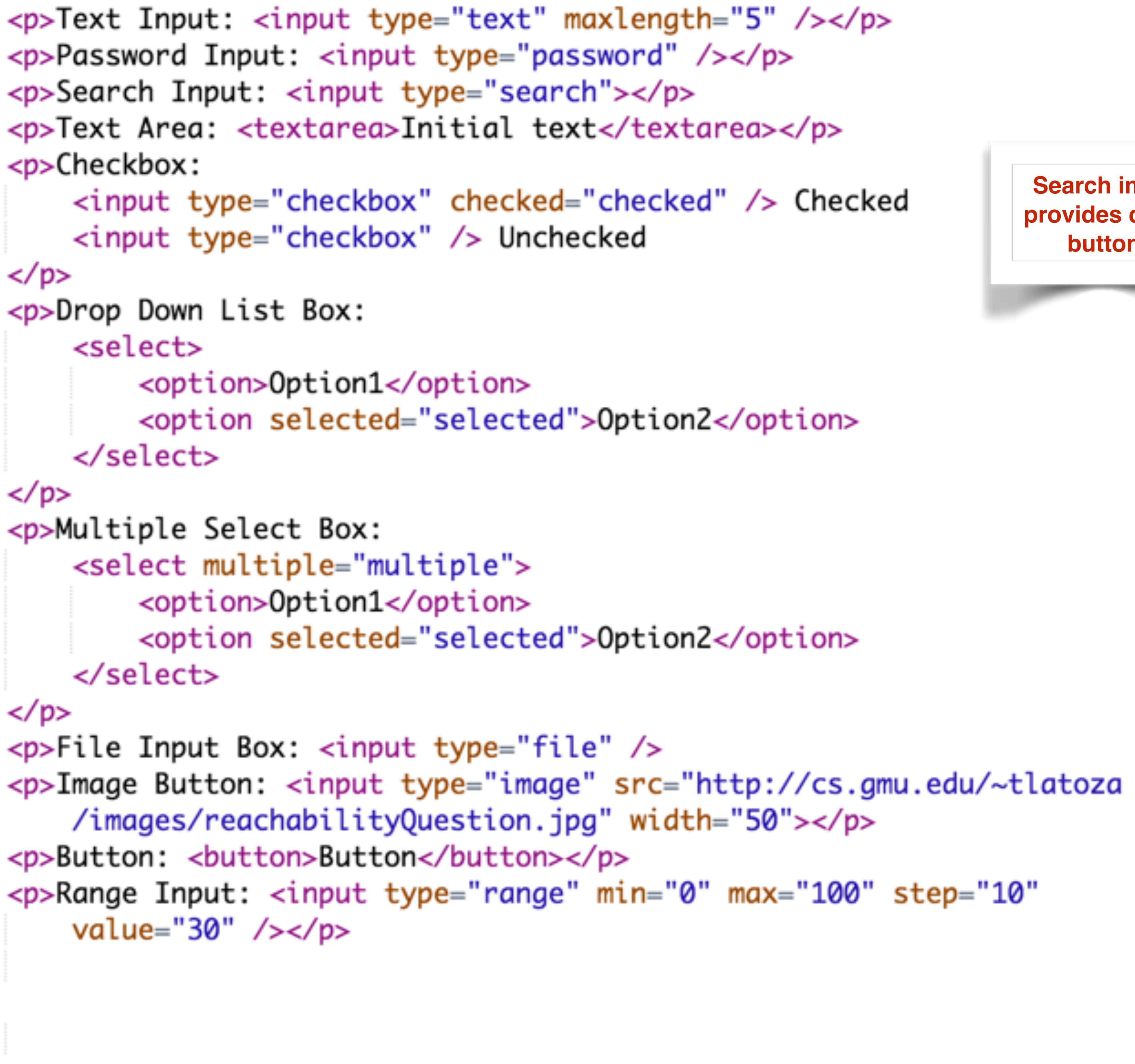
"Send form data in HTTP headers"

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

Method specifies how data is transmitted to server. method="get" sends data appended to URL

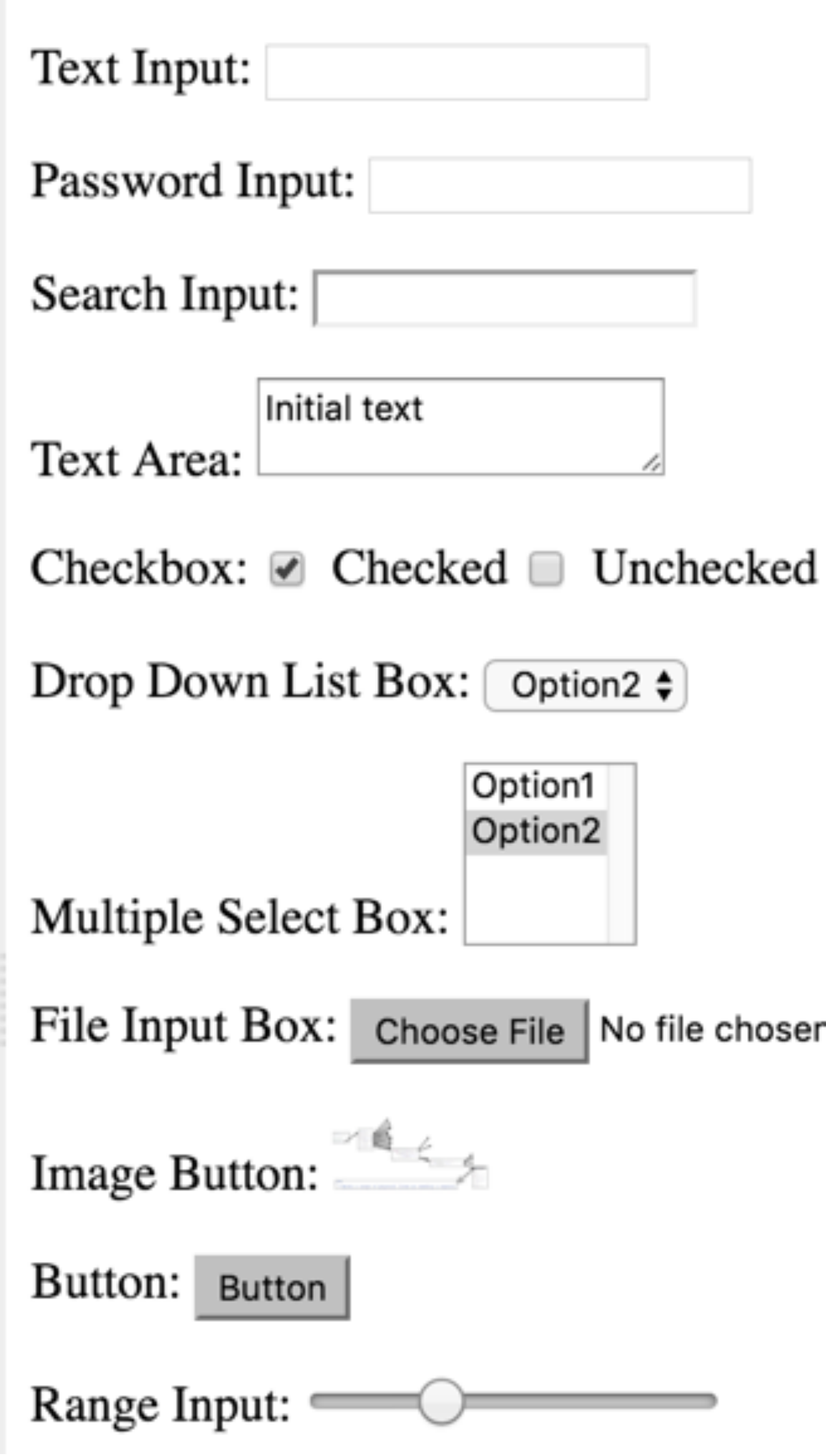
- Elements located in a form may have *name* and *value* attributes. This data is used in submission to server.
    - Note: name is used for a very different purpose than id.
- Controls may (or may not) be enclosed in a form.
    - If not form submission mechanism to submit data to server, no need for form.

Controls



```
<p>Text Input: <input type="text" maxlength="5" /></p>
<p>Password Input: <input type="password" /></p>
<p>Search Input: <input type="search"></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="selected">Option2</option>
    </select>
</p>
<p>Multiple Select Box:
    <select multiple="multiple">
        <option>Option1</option>
        <option selected="selected">Option2</option>
    </select>
</p>
<p>File Input Box: <input type="file" />
<p>Image Button: <input type="image" src="http://cs.gmu.edu/~tlatoza
    /images/reachabilityQuestion.jpg" width="50"></p>
<p>Button: <button>Button</button></p>
<p>Range Input: <input type="range" min="0" max="100" step="10"
    value="30" /></p>
```

Search input provides clear button



Text Input:

Password Input:

Search Input:

Text Area: Initial text

Checkbox: ☑ Checked ☐ Unchecked

Drop Down List Box: Option2

Multiple Select Box: Option1 Option2

File Input Box: Choose File No file choser

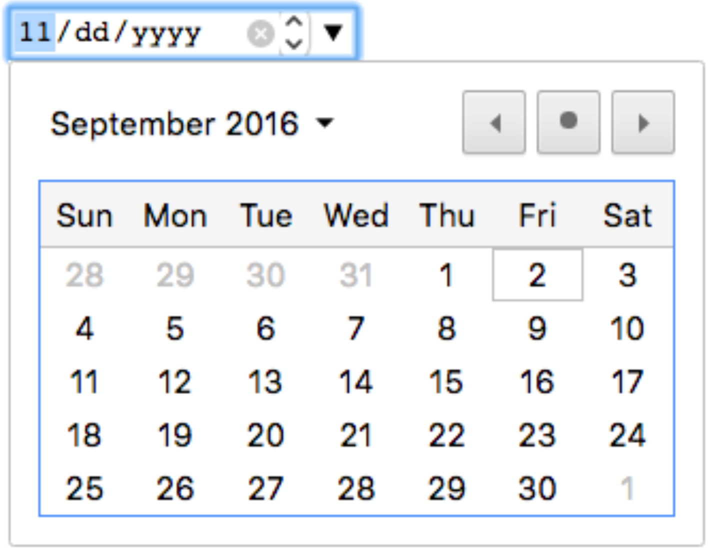
Image Button:

Button: Button

Range Input:

Specialized controls

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

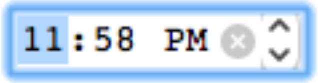


11/dd/yyyy

September 2016

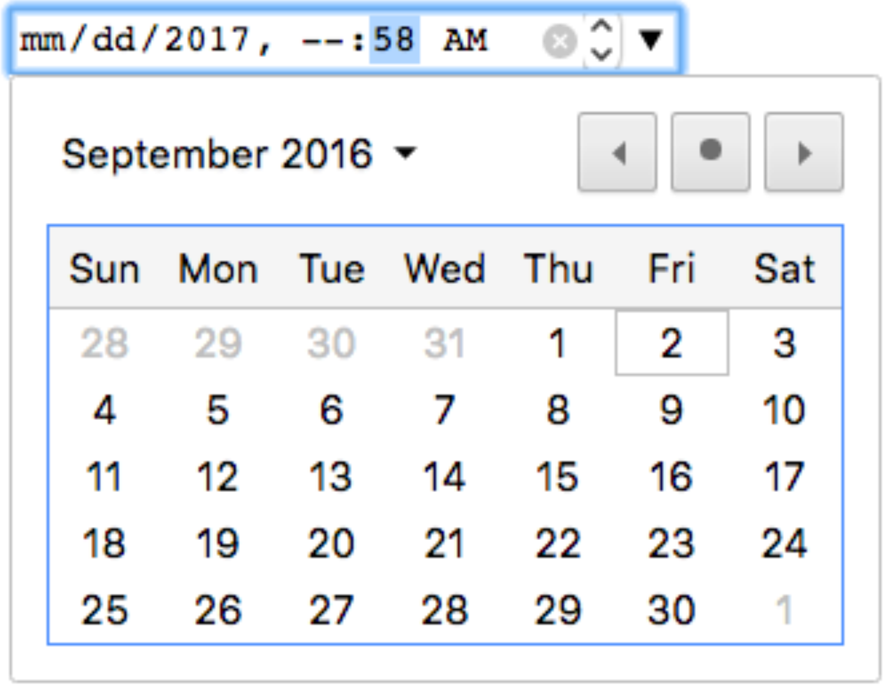
Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1

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



11:58 PM

```
<input type="datetime-local" />
```

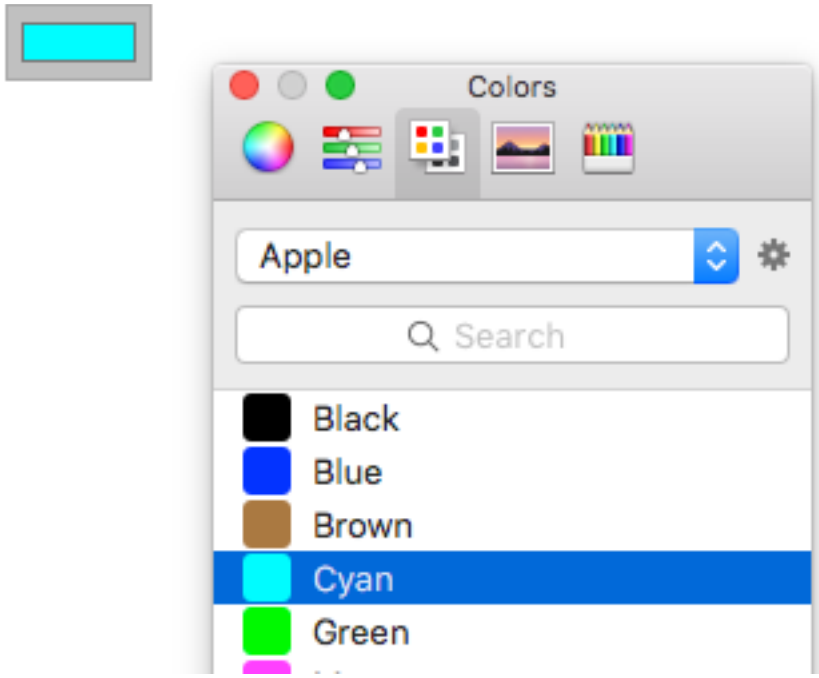


mm/dd/2017, --:58 AM

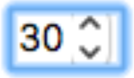
September 2016

Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1

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



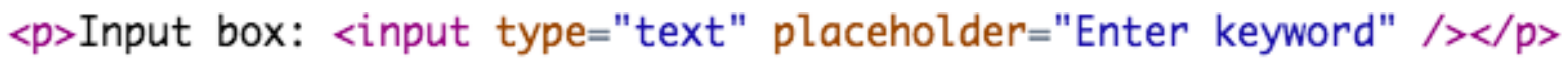
```
<input type="number" min="0" max="50"/>
```



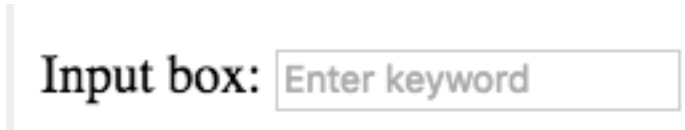
30

Labeling input

- Can place suggested input or prompt *inside* input element

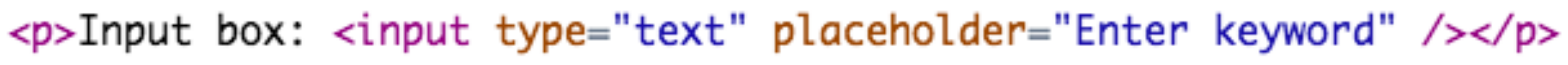


```
<p>Input box: <input type="text" placeholder="Enter keyword" /></p>
```

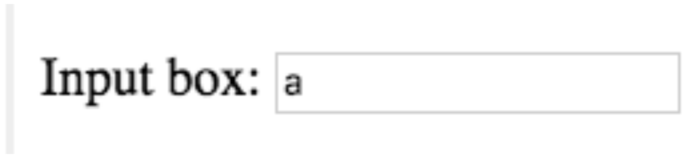


Input box: Enter keyword

  - Disappears after user types

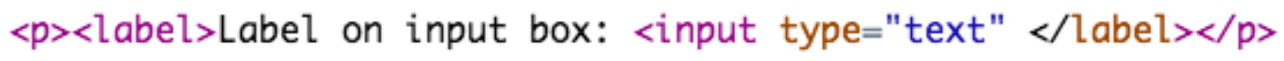


```
<p>Input box: <input type="text" placeholder="Enter keyword" /></p>
```

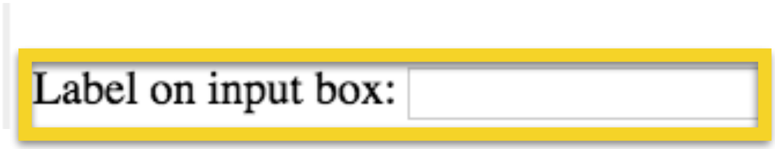


Input box: a

- Label attaches a label *and* expands the clickable region of control, making form easier to use



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

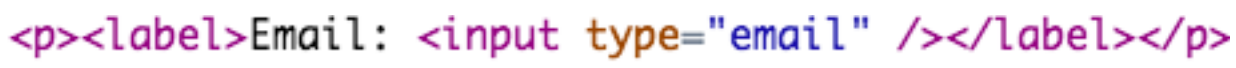


Label on input box:

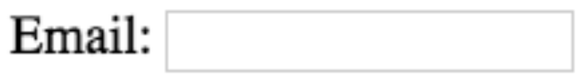
Clickable region

Validating input

- Displays errors on invalid input *immediately*, making it easier to fix errors
- Check that input is a valid email

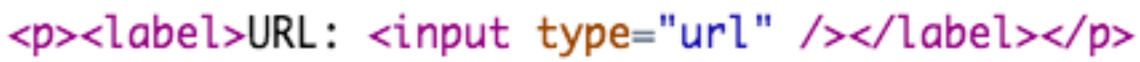


```
<p><label>Email: <input type="email" /></label></p>
```

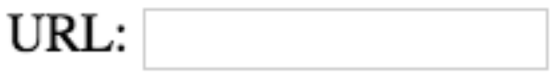


Email:

- Check that input is a valid URL

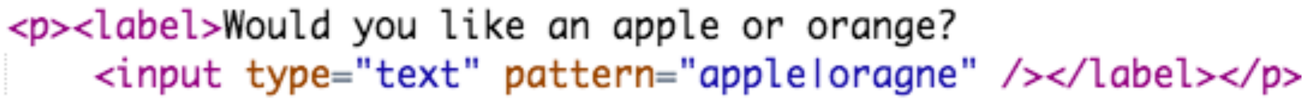


```
<p><label>URL: <input type="url" /></label></p>
```

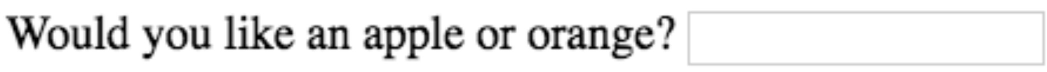


URL:

- Check that input matches regex pattern

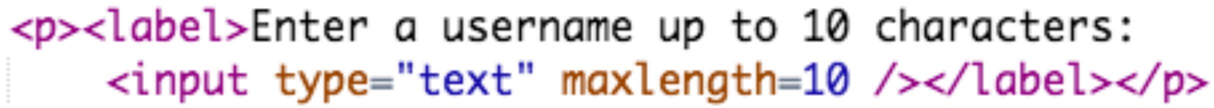


```
<p><label>Would you like an apple or orange?
    <input type="text" pattern="apple|oragne" /></label></p>
```

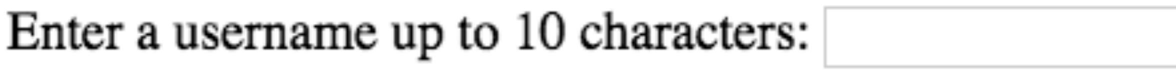


Would you like an apple or orange?

- Constrain input to be at most maxlength

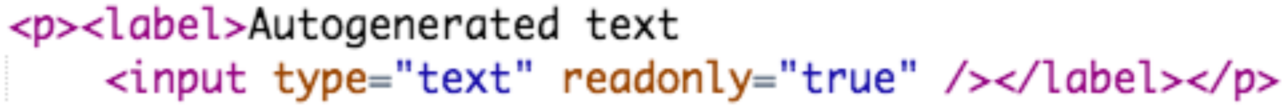


```
<p><label>Enter a username up to 10 characters:
    <input type="text" maxlength=10 /></label></p>
```

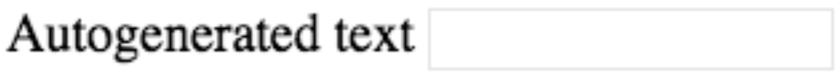


Enter a username up to 10 characters:

- Prevent all edits



```
<p><label>Autogenerated text
    <input type="text" readonly="true" /></label></p>
```

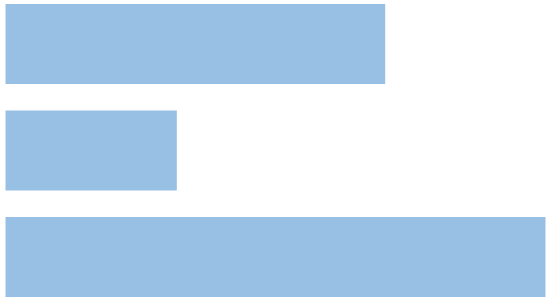


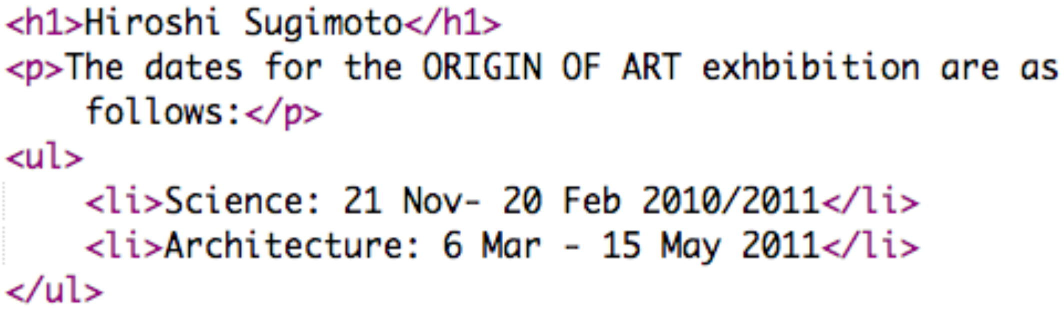
Autogenerated text

Block vs. Inline Elements

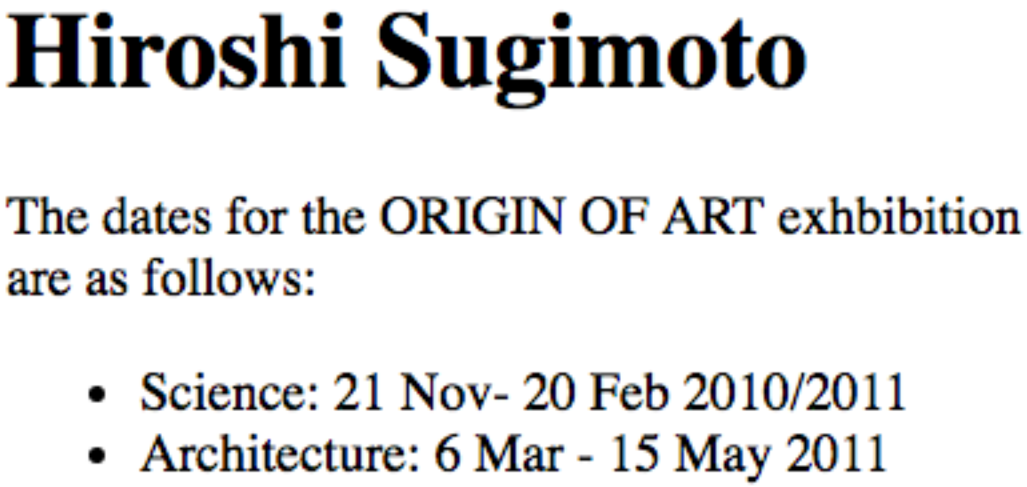
Block elements

Block elements appear on a new line.
Examples: <h1><p><li><table><form>





```
<h1>Hiroshi Sugimoto</h1>
<p>The dates for the ORIGIN OF ART exhbition are as
   follows:</p>
<ul>
   <li>Science: 21 Nov- 20 Feb 2010/2011</li>
   <li>Architecture: 6 Mar - 15 May 2011</li>
</ul>
```



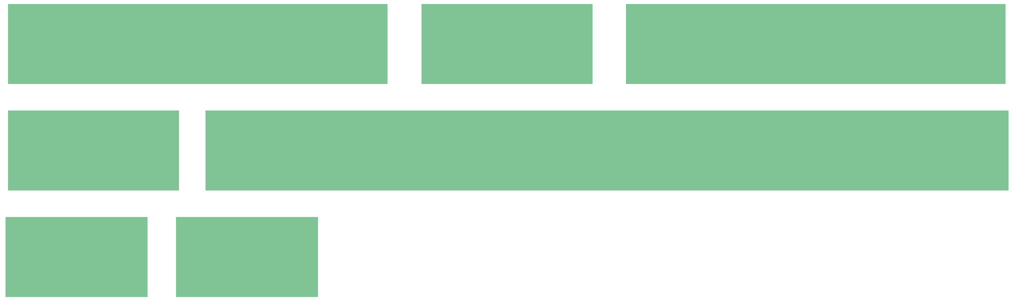
Hiroshi Sugimoto

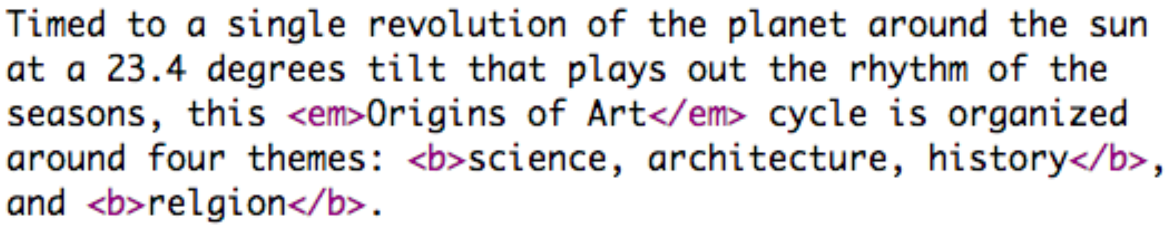
The dates for the ORIGIN OF ART exhbition are as follows:

- Science: 21 Nov- 20 Feb 2010/2011
- Architecture: 6 Mar - 15 May 2011

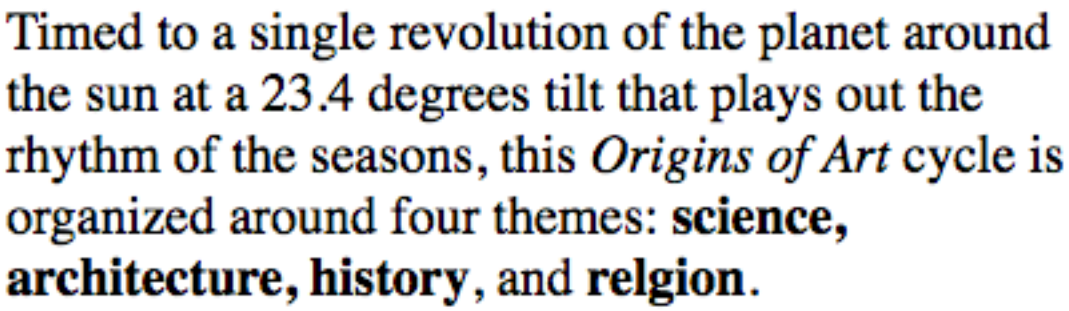
Inline elements

Inline elements appear to continue on the same line.
Examples: <a><b><input><img>





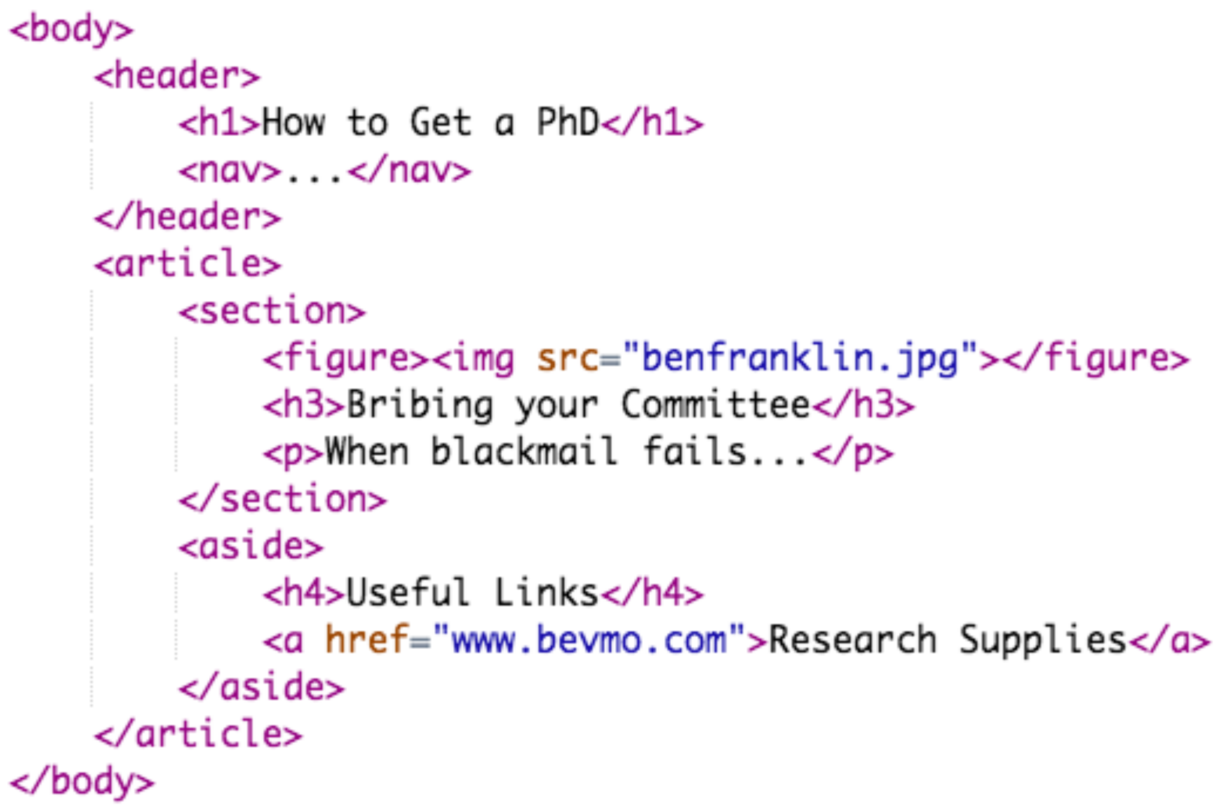
```
Timed to a single revolution of the planet around the sun
at a 23.4 degrees tilt that plays out the rhythm of the
seasons, this <em>Origins of Art</em> cycle is organized
around four themes: <b>science, architecture, history</b>,
and <b>relgion</b>.
```



Timed to a single revolution of the planet around the sun at a 23.4 degrees tilt that plays out the rhythm of the seasons, this *Origins of Art* cycle is organized around four themes: **science, architecture, history**, and **relgion**.

Grouping elements

- Creates a **parent** or **container** element and a set of **child** elements
- Enables group to be styled together
- Can use any block or inline element or *generic* element
    - **<div>** is the generic block element
    - **<span>** is the generic inline element
- Semantic layout elements are block elements that associate meaning with group
    - Very useful for CSS selectors (coming soon)



```
<body>
    <header>
        <h1>How to Get a PhD</h1>
        <nav>...</nav>
    </header>
    <article>
        <section>
            <figure><img src="benfranklin.jpg"></figure>
            <h3>Bribing your Committee</h3>
            <p>When blackmail fails...</p>
        </section>
        <aside>
            <h4>Useful Links</h4>
            <a href="www.bevmo.com">Research Supplies</a>
        </aside>
    </article>
</body>
```

Some popular semantic layout elements
<header><footer><nav><article><aside>
<section><figcaption>

HTML Style

- Tags
  - Use lowercase for names
  - Use indentation to reflect hierarchy
  - Always close tags
    - Or use self-closing tags <tagname /> notation
- Use attributename=“value” format for attributes
- Use blank lines to break up documents into closely connected regions
- Use comments to describe purpose of regions

HTML Best Practices

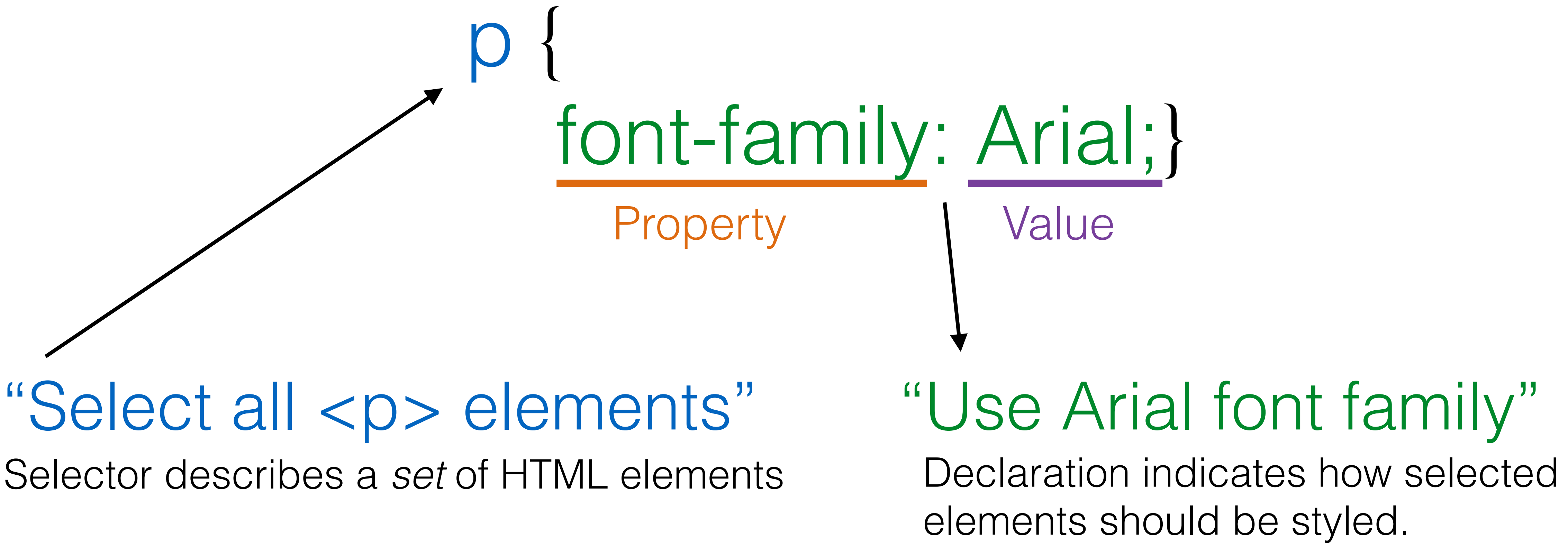
- Use specialized controls or input validation where applicable
- Always include elements of HTML starter document
- Use label or placeholder for labeling controls
- Use alt to make images accessible

In Class Exercise

- Form two person groups.
  - Build a simple personal website.
  - It should use basic HTML elements such as <img>, <a>, <div>, semantic tags.
  - Use a pastebin such as seecode.run or jsbin.com
- When you are done, log in to Socrative, post link to your pastebin.

CSS: Cascading Style Sheets

- Language for *styling* documents



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

Property

Value

“Select all <p> elements”

Selector describes a *set* of HTML elements

“Use Arial font family”

Declaration indicates how selected elements should be styled.

- Separates **visual presentation** (CSS) from **document structure** (HTML)
  - Enables changes to one or the other.
  - Enables styles to be *reused* across sets of elements.

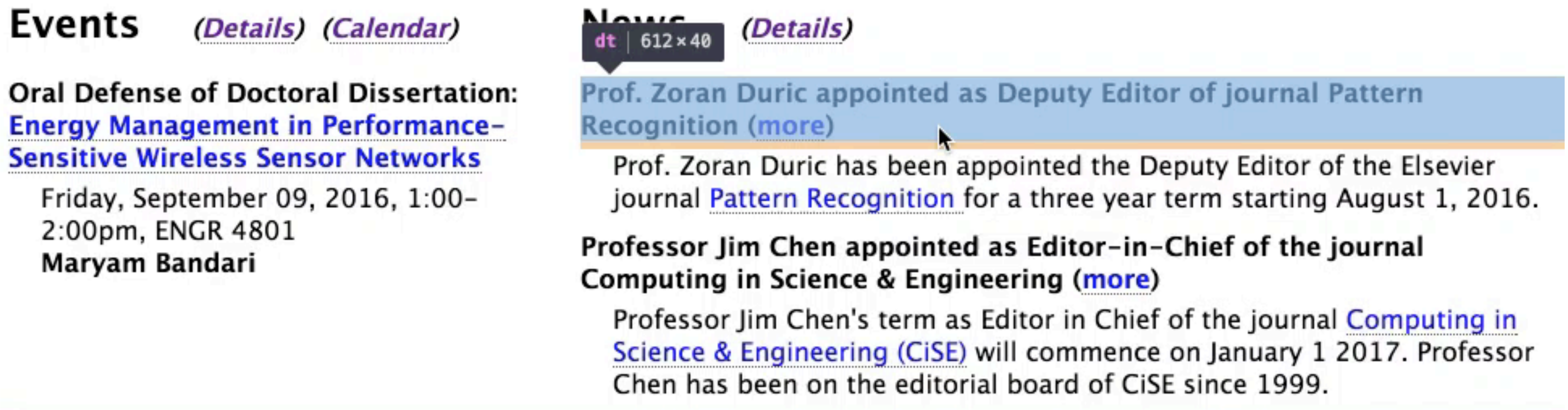
CSS History

- 1994: Cascading HTML style sheets—a proposal
  - Hakon W Lie proposes CSS
  - Working w/ Tim-Berners Lee at CERN
- 1996: CSS1 standard, recommended by W3C
  - Defines basic styling elements like font, color, alignment, margin, padding, etc.
- 1998: CSS2 standard, recommended by W3C
  - Adds positioning schemes, z-index, new font properties
- 2011: CSS3 standards divided into modules, begin adoption
  - Add more powerful selectors, more powerful attributes

https://dev.opera.com/articles/css-twenty-years-hakon/

https://en.wikipedia.org/wiki/Cascading_Style_Sheets#History

CSS Styling



Events *(Details)* *(Calendar)*

Oral Defense of Doctoral Dissertation:
Energy Management in Performance-Sensitive Wireless Sensor Networks
Friday, September 09, 2016, 1:00-2:00pm, ENGR 4801
Maryam Bandari

News *(Details)*

Prof. Zoran Duric appointed as Deputy Editor of journal Pattern Recognition (more)

Prof. Zoran Duric has been appointed the Deputy Editor of the Elsevier journal Pattern Recognition for a three year term starting August 1, 2016.

Professor Jim Chen appointed as Editor-in-Chief of the journal Computing in Science & Engineering (more)

Professor Jim Chen's term as Editor in Chief of the journal Computing in Science & Engineering (CiSE) will commence on January 1 2017. Professor Chen has been on the editorial board of CiSE since 1999.

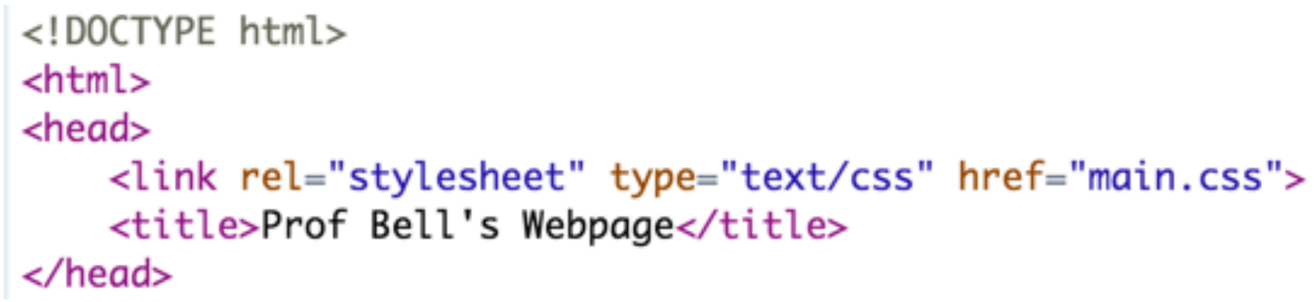
- Invisible box around every element.
- Rules control how sets of boxes and their contents are presented

Example Styles

BOXES	TEXT
Width, height	Typeface
Borders (color, width, style)	Size, color
Position in the browser window	Italics, bold, lowercase

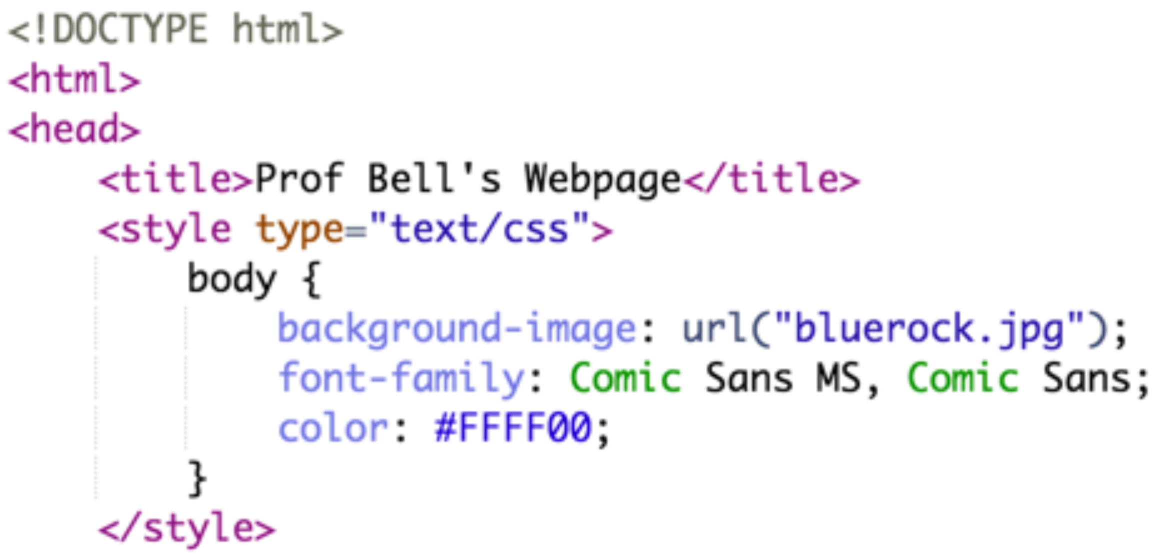
Using CSS

External CSS



```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="main.css">
    <title>Prof Bell's Webpage</title>
</head>
```

Internal CSS



```
<!DOCTYPE html>
<html>
<head>
    <title>Prof Bell's Webpage</title>
    <style type="text/css">
        body {
            background-image: url("bluerock.jpg");
            font-family: Comic Sans MS, Comic Sans;
            color: #FFFF00;
        }
    </style>
```

- External CSS enables stylesheets to be reused across *multiple* files
- Can include CSS files
- Can nest CSS files
    - @import url("file.css") imports a CSS file in a CSS file

CSS Type Selectors

- What if we wanted more green?

```
h2, h3 {
    color: LightGreen;
}
```

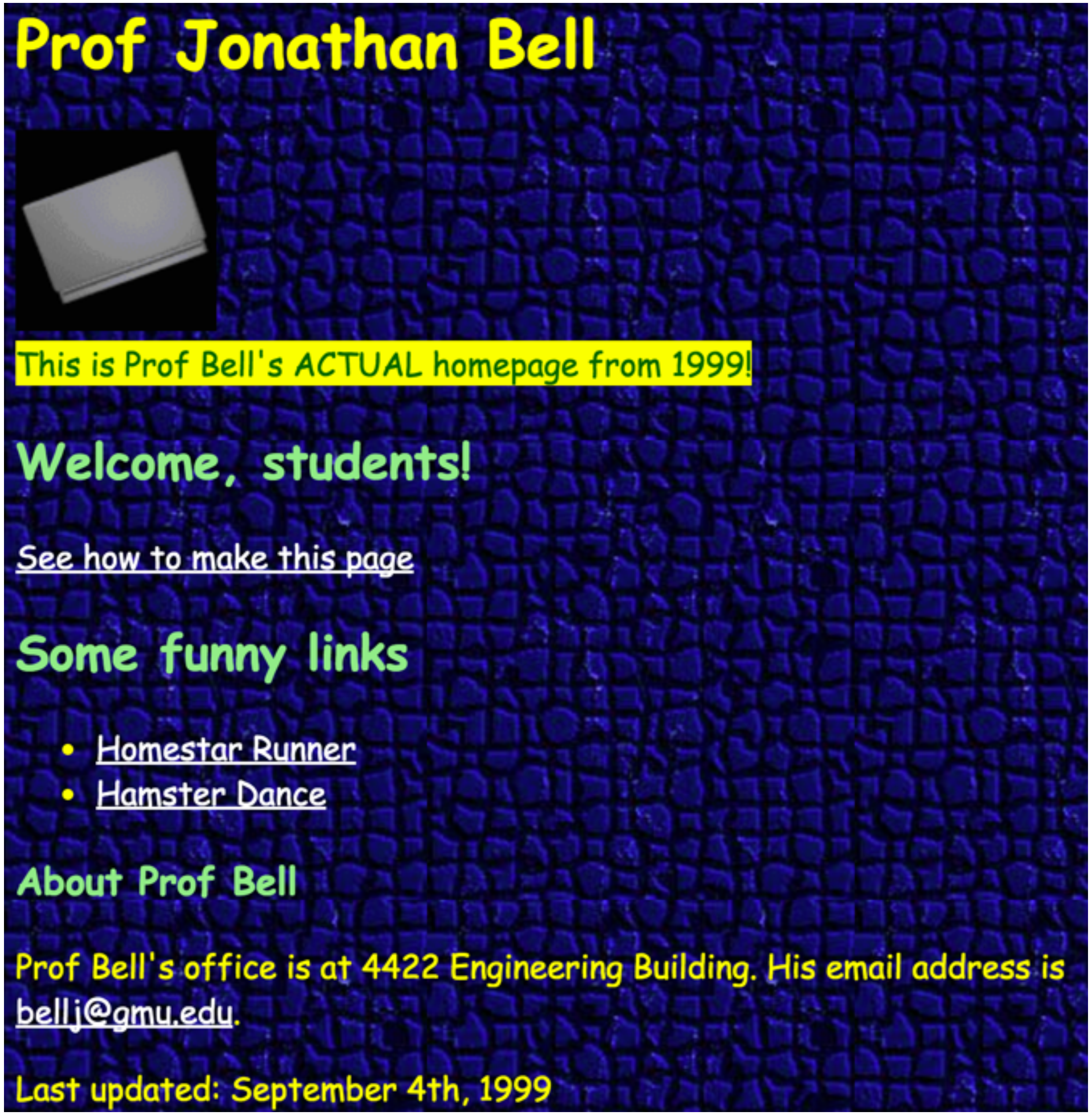
“Select all <h2> and <h3> elements”

Type selector selects one or more element types.

```
* {
    color: LightGreen;
}
```

“Select all elements”

Universal selector selects all elements.



Prof Jonathan Bell

This is Prof Bell's ACTUAL homepage from 1999!

Welcome, students!

See how to make this page

Some funny links

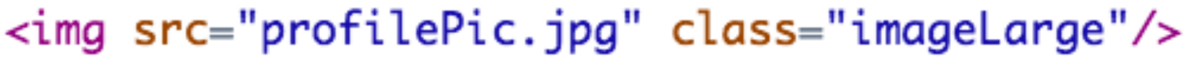
- Homestar Runner
- Hamster Dance

About Prof Bell

Prof Bell's office is at 4422 Engineering Building. His email address is bellj@gmu.edu.

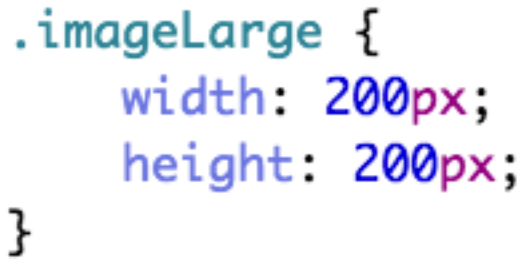
Last updated: September 4th, 1999

CSS Class Selectors



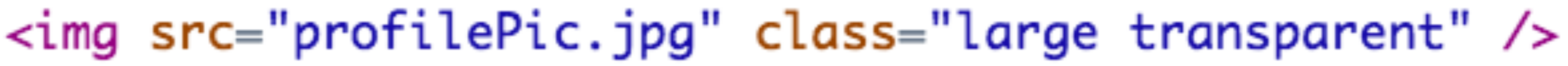
```
<img src="profilePic.jpg" class="imageLarge"/>
```

"Label <img> element with imageLarge class"

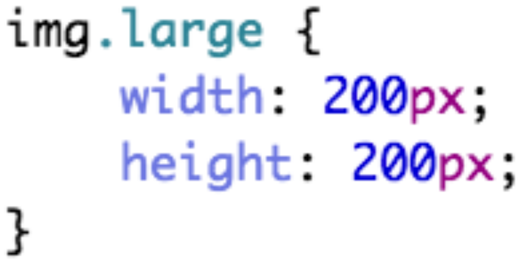


```
.imageLarge {
    width: 200px;
    height: 200px;
}
```

"Define class imageLarge."

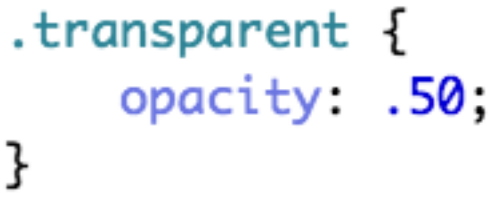


```
<img src="profilePic.jpg" class="large transparent" />
```



```
img.large {
    width: 200px;
    height: 200px;
}
```

"Define large class that applies only to <img> elements"

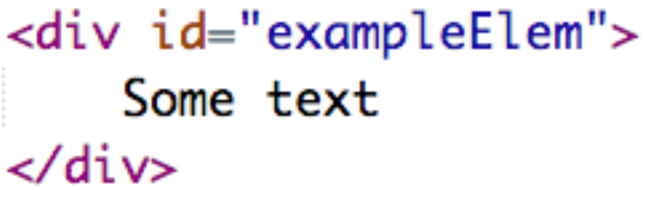


```
.transparent {
    opacity: .50;
}
```

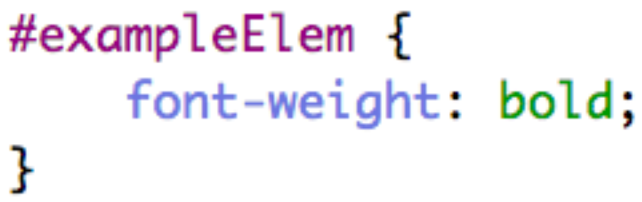
"Define transparent class"

- Classes enable the creation of sets of elements that can be styled in the same way.

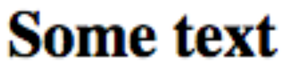
CSS id selectors



```
<div id="exampleElem">
    Some text
</div>
```



```
#exampleElem {
    font-weight: bold;
}
```



Some text

- Advantages
    - Control presentation of individual elements
- Disadvantages
    - Must write separate rule for *each* element

Additional selector types

Selector	Meaning	Example	
Descendant selector	Matches all descendants of an element	p a { }	Select <a> elements inside <p> elements
Child selector	Matches a direct child of an element	h1>a { }	Select <a> elements that are directly contained by <h1> elements.
First child selector	Matches the first child of an element	h1:first-child { }	Select the the elements that are the first child of a <h1> element.
Adjacent selector	Matches selector	h1+p { }	Selects the first <p> element after any <h1> element
Negation selector	Selects all elements that are not selected.	body *:not(p)	Select all elements in the body that are not <p> elements.
Attribute selector	Selects all elements that define a specific attribute.	input[invalid]	Select all <input> elements that have the invalid attribute.
Equality attribute selector	Select all elements with a specific attribute value	p[class="invisible"]	Select all <p> elements that have the invisible class.

CSS Selectors

- Key principles in designing effective styling rules
  - Use classes, semantic tags to create sets of elements that share a similar rules
  - Don’t repeat yourself (DRY)
    - Rather than create many identical or similar rules, apply single rule to all similar elements
  - Match based on semantic properties, not styling
    - Matching elements based on their pre-existing styling is **fragile**

Cascading selectors

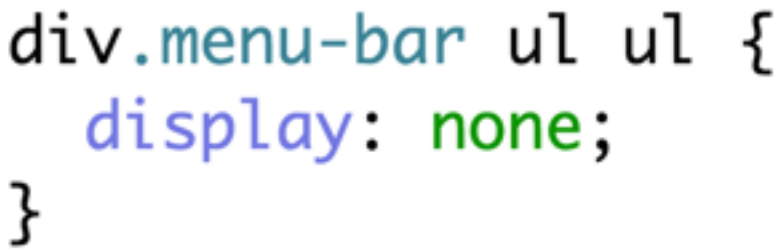
- What happens if more than one rule applies?
- Most *specific* rule takes precedence
    - p b is more specific than p
    - #maximizeButton is more specific than button
- If otherwise the same, *last* rule wins
- Enables writing generic rules that apply to many elements that are overriden by specific rules applying to a few elements

CSS inheritance

- When an element is contained inside another element, some styling properties are inherited
  - e.g., font-family, color
- Some properties are not inherited
  - e.g., background-color, border
- Can force many properties to inherit value from parent using the inherit value
  - e.g., padding: inherit;

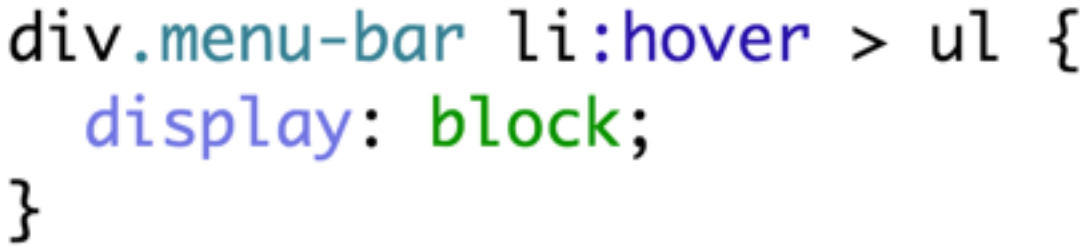
Exercise - What is selected?

1.



```
div.menu-bar ul ul {
  display: none;
}
```

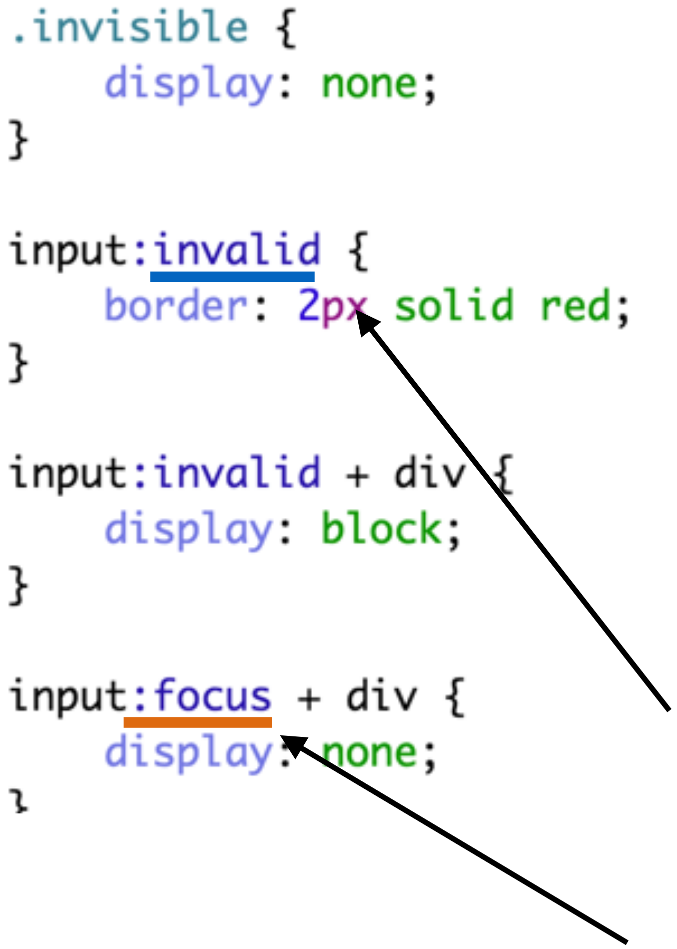
2.



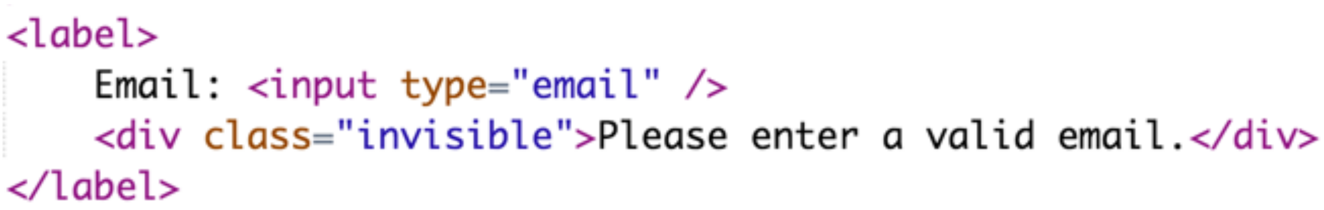
```
div.menu-bar li:hover > ul {
  display: block;
}
```

ul: unordered list
li: list element

Pseudo classes



```
.invisible {
    display: none;
}

input:invalid {
    border: 2px solid red;
}

input:invalid + div {
    display: block;
}

input:focus + div {
    display: none;
}
```



```
<label>
    Email: <input type="email" />
    <div class="invisible">Please enter a valid email.</div>
</label>
```

“Select elements with the invalid attribute.”

“Select elements that have focus.”

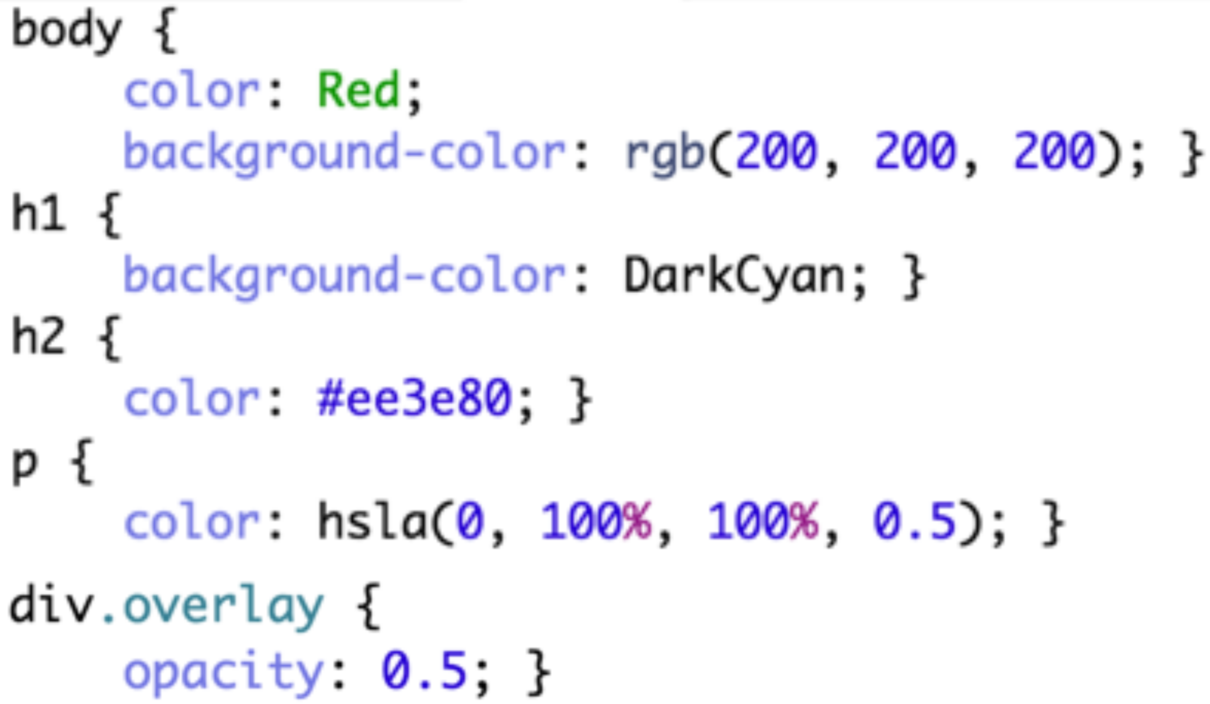
- Classes that are automatically attached to elements based on their attributes.

Examples of pseudo classes

- :active - elements activated by user. For mouse clicks, occurs between mouse down and mouse up.
- :checked - radio, checkbox, option elements that are checked by user
- :disabled - elements that can't receive focus
- :empty - elements with no children
- :focus - element that currently has the focus
- :hover - elements that are currently hovered over by mouse
- :invalid - elements that are currently invalid
- :link - link element that has not yet been visited
- :visited - link element that has been visited

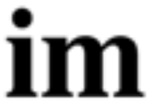
Color

- Can set text color (color) and background color (background-color)
- Several ways to describe color
  - six digit hex code (e.g., #ee3e80)
  - color names: 147 predefined names
  - rgb(red, green, blue): amount of red, green, and blue
  - hsla(hue, saturation, lightness, alpha): alternative scheme for describing colors
- Can set opacity (opacity) from 0.0 to 1.0

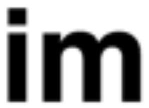


```
body {
    color: Red;
    background-color: rgb(200, 200, 200); }
h1 {
    background-color: DarkCyan; }
h2 {
    color: #ee3e80; }
p {
    color: hsla(0, 100%, 100%, 0.5); }
div.overlay {
    opacity: 0.5; }
```

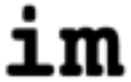
Typefaces



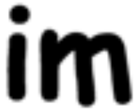
Serif



Sans-Serif



Monospace



Cursive

font-family: Georgia, Times, serif;

"Use Georgia if available, otherwise Times, otherwise any serif font".

font-family enables the typeface to be specified. The typeface must be installed. Lists of fonts enable a browser to select an alternative.

Type scales

12 pixel scale

Pixels			Percentages	
h1	24px		h1	200%
h2	18px	=	h2	150%
h3	14px		h3	117%
body	12px		body	75%

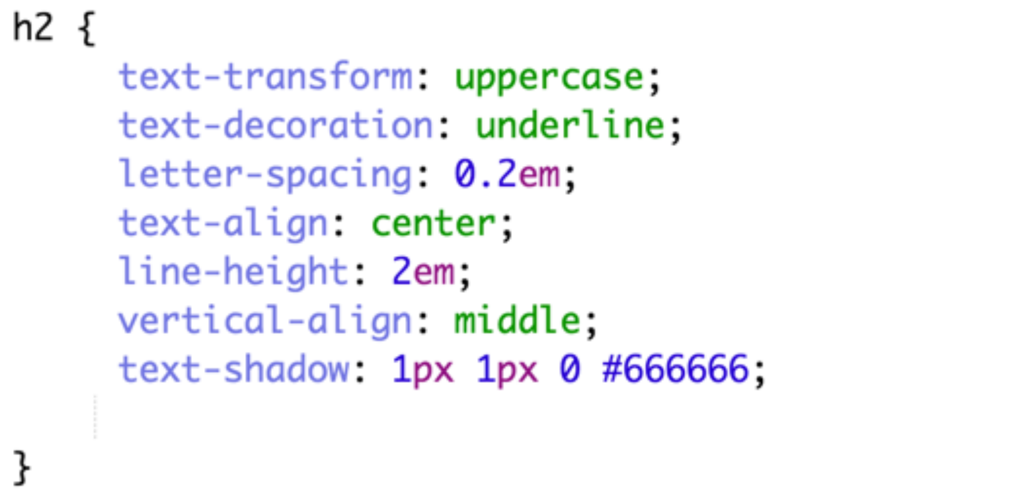
16 pixel scale

Pixels			Percentages	
h1	32px		h1	200%
h2	24px	=	h2	150%
h3	18px		h3	112.5%
body	16px		body	100%

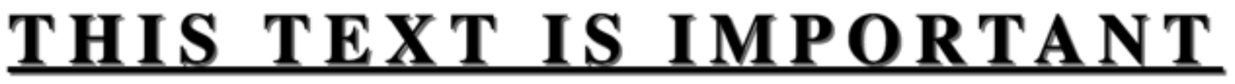
Sets absolute font size.

Sets font size relative to default text size (16px)

Styling text



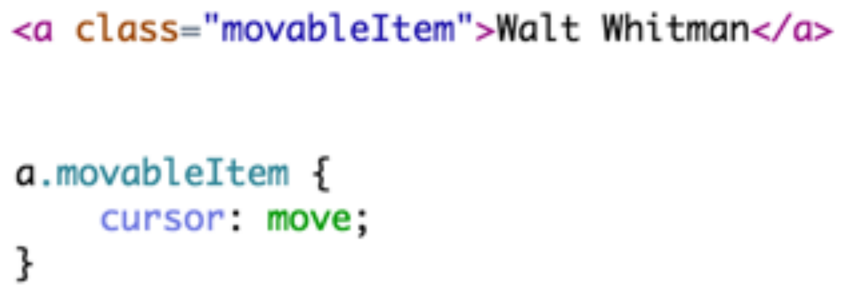
```
h2 {
    text-transform: uppercase;
    text-decoration: underline;
    letter-spacing: 0.2em;
    text-align: center;
    line-height: 2em;
    vertical-align: middle;
    text-shadow: 1px 1px 0 #666666;

}
```



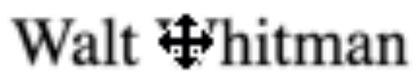
THIS TEXT IS IMPORTANT

- text-transform: uppercase, lowercase, capitalize
- text-decoration: none, underline, overline, line-through, blink
- letter-spacing: space between letters (kerning)
- text-align: left, right, center, justify
- line-height: total of font height and empty space between lines
- vertical-align: top, middle, bottom, …
- text-shadow: [x offset][y offset][blur offset][color]

Cursor



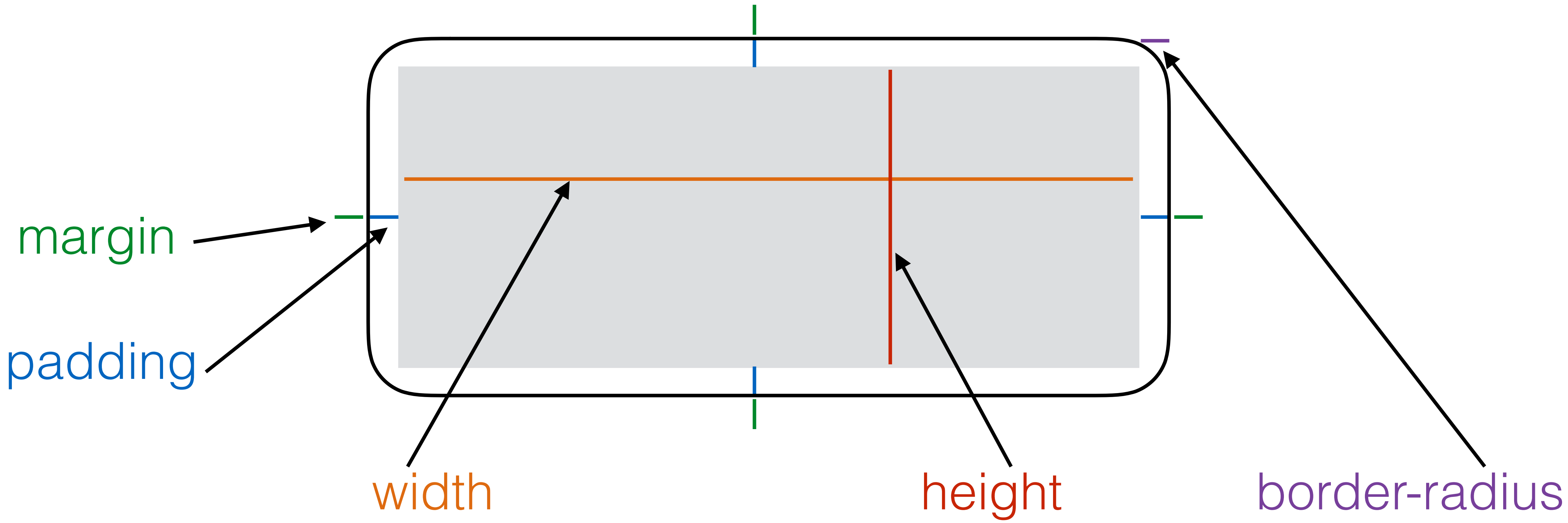
```
<a class="movableItem">Walt Whitman</a>


a.movableItem {
    cursor: move;
}
```



Walt Whitman

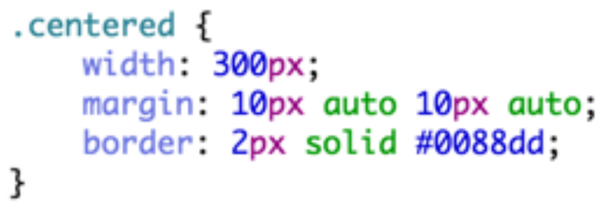
- Can change the default cursor with cursor attribute
  - auto, crosshair, pointer, move, text, wait, help, url(“cursor.gif”)
- Should *only* do this if action being taken clearly matches cursor type

Box properties

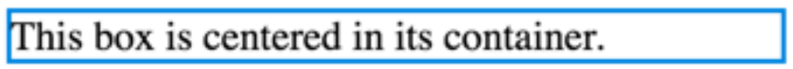


- Boxes, by default, are sized *just* large enough to fit their contents.
- Can specify sizes using px or %
    - % values are relative to the container dimensions
- margin: 10px 5px 10px 5px; (clockwise order - [top] [right] [bottom] [left])
- border: 3px dotted #0088dd; ([width] [style] [color])
    - style may be solid, dotted,dashed, double, groove, ridge, inset, outset, hidden / none

Centering content



```
.centered {
    width: 300px;
    margin: 10px auto 10px auto;
    border: 2px solid #0088dd;
}
```

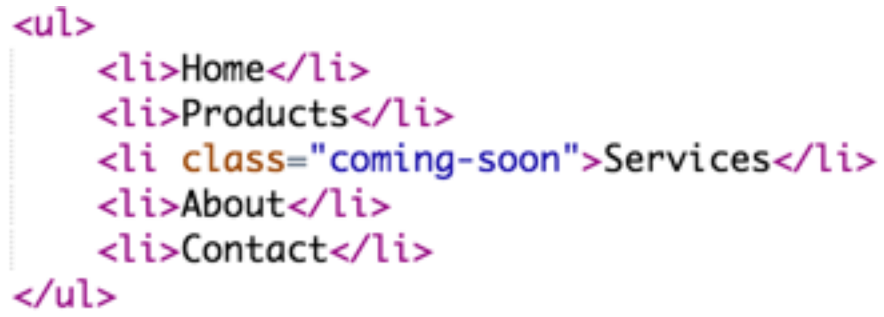


This box is centered in its container.

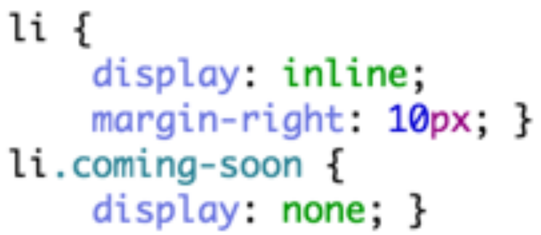
- How do you center an element inside a container?
- Step 1: Must first ensure that element is *narrower* than container.
  - By default, element will expand to fill entire container.
  - So must usually explicitly set width for element.
- Step 2: Use *auto* value for left and right to create equal gaps

Visibility and layout

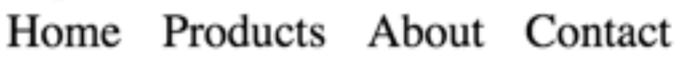
- Can force elements to be inline or block element.
  - display: inline
  - display: block
- Can cause element to not be laid out or take up any space
  - display: none
  - *Very* useful for content that is dynamically added and removed.
- Can cause boxes to be invisible, but still take up space
  - visibility: hidden;



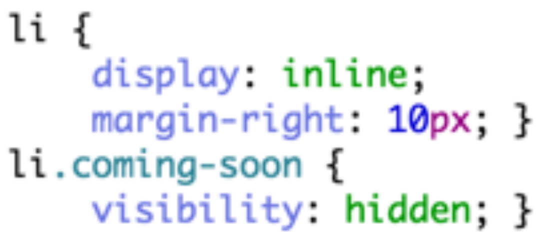
```
<ul>
    <li>Home</li>
    <li>Products</li>
    <li class="coming-soon">Services</li>
    <li>About</li>
    <li>Contact</li>
</ul>
```



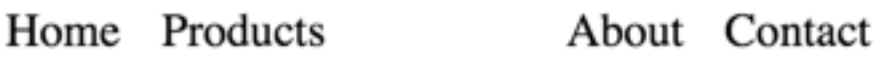
```
li {
    display: inline;
    margin-right: 10px; }
li.coming-soon {
    display: none; }
```



```
Home  Products  About  Contact
```



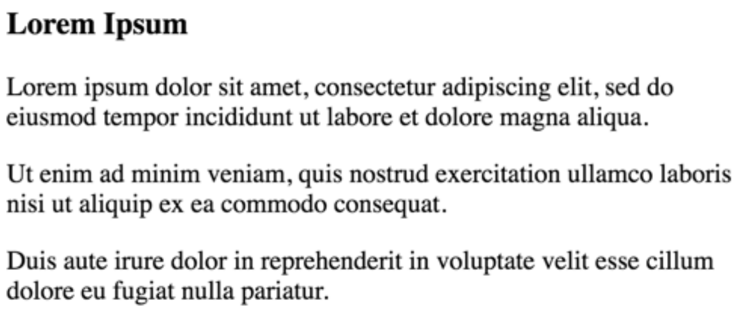
```
li {
    display: inline;
    margin-right: 10px; }
li.coming-soon {
    visibility: hidden; }
```



```
Home  Products             About  Contact
```

Positioning schemes

Normal flow (default)



Lorem Ipsum

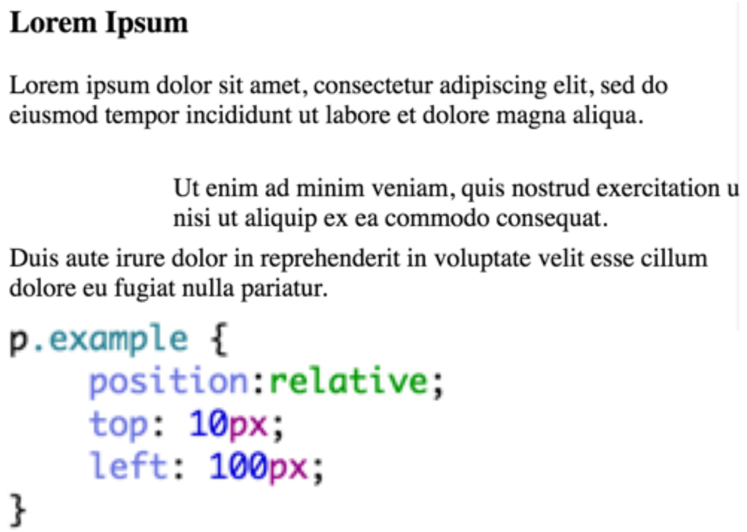
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Block level elements appear on a new line. Even if there is space, boxes will not appear next to each other.

Relative positioning



Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

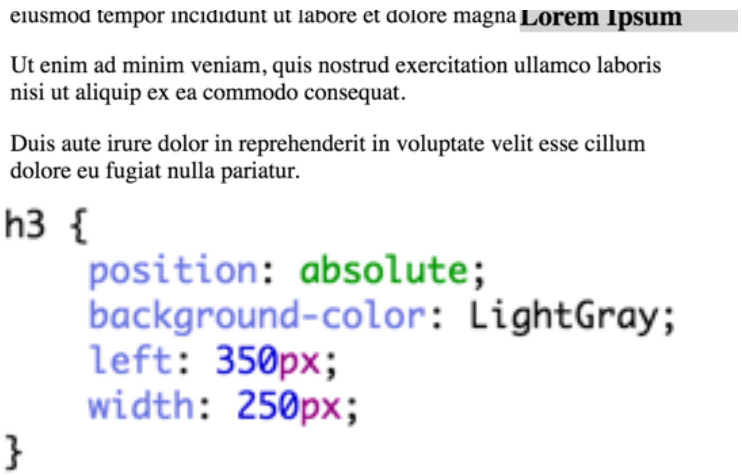
Ut enim ad minim veniam, quis nostrud exercitation u
nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

```
p.example {
    position:relative;
    top: 10px;
    left: 100px;
}
```

Element shifted from normal flow. Position of other elements is *not* affected.

Absolute positioning



eiusmod tempor incididunt ut labore et dolore magna **Lorem Ipsum**

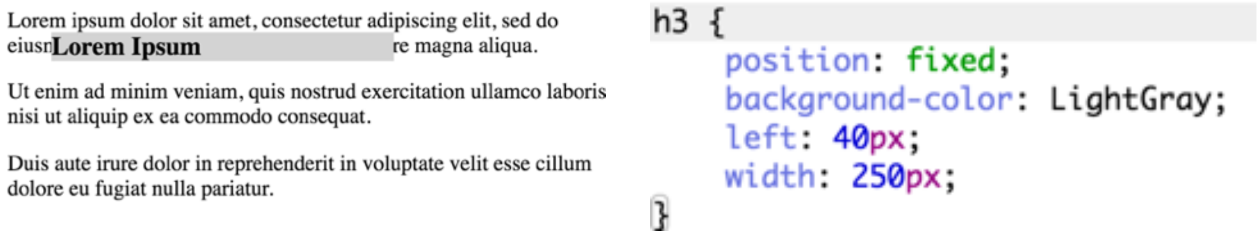
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

```
h3 {
    position: absolute;
    background-color: LightGray;
    left: 350px;
    width: 250px;
}
```

Element taken out of normal flow and does not affect position of other elements. Moves as user scrolls.

Fixed positioning



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusn**Lorem Ipsum** re magna aliqua.

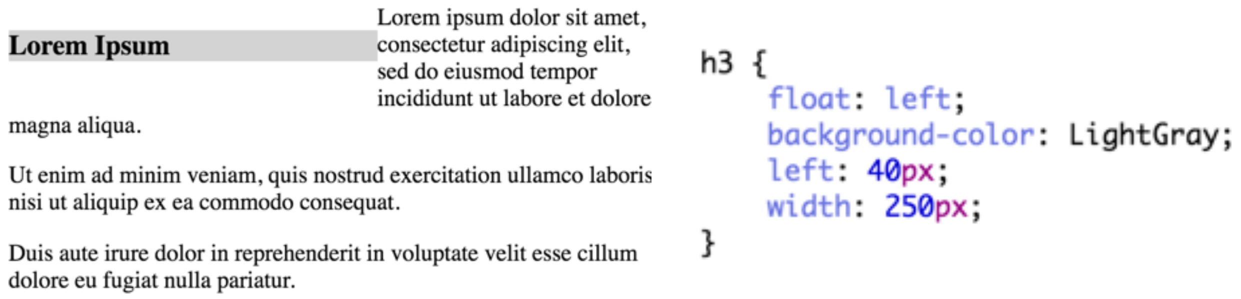
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

```
h3 {
    position: fixed;
    background-color: LightGray;
    left: 40px;
    width: 250px;
}
```

Element taken out of normal flow and does not affect position of other elements. Fixed in window position as user scrolls.

Floating elements



Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

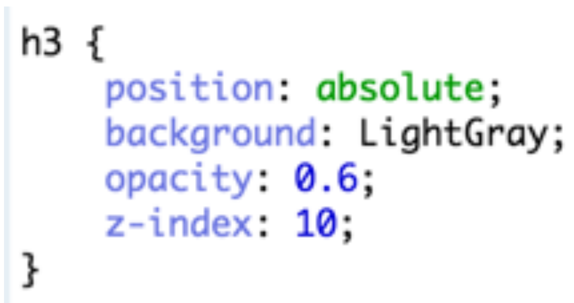
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

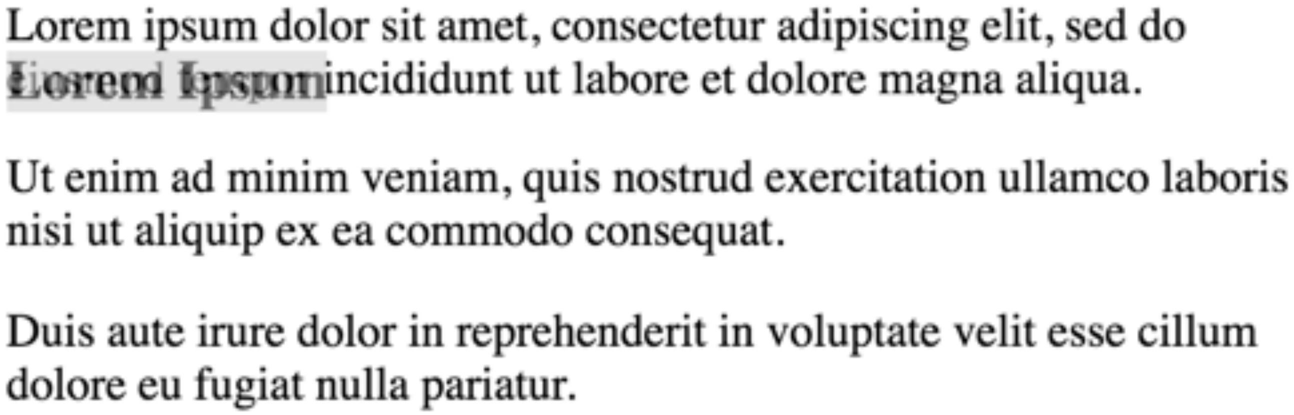
```
h3 {
    float: left;
    background-color: LightGray;
    left: 40px;
    width: 250px;
}
```

Element taken out of normal flow and position to far left or right of container. Element becomes block element that others flow around.

Stacking elements



```
h3 {
    position: absolute;
    background: LightGray;
    opacity: 0.6;
    z-index: 10;
}
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do

Lorem Ipsum

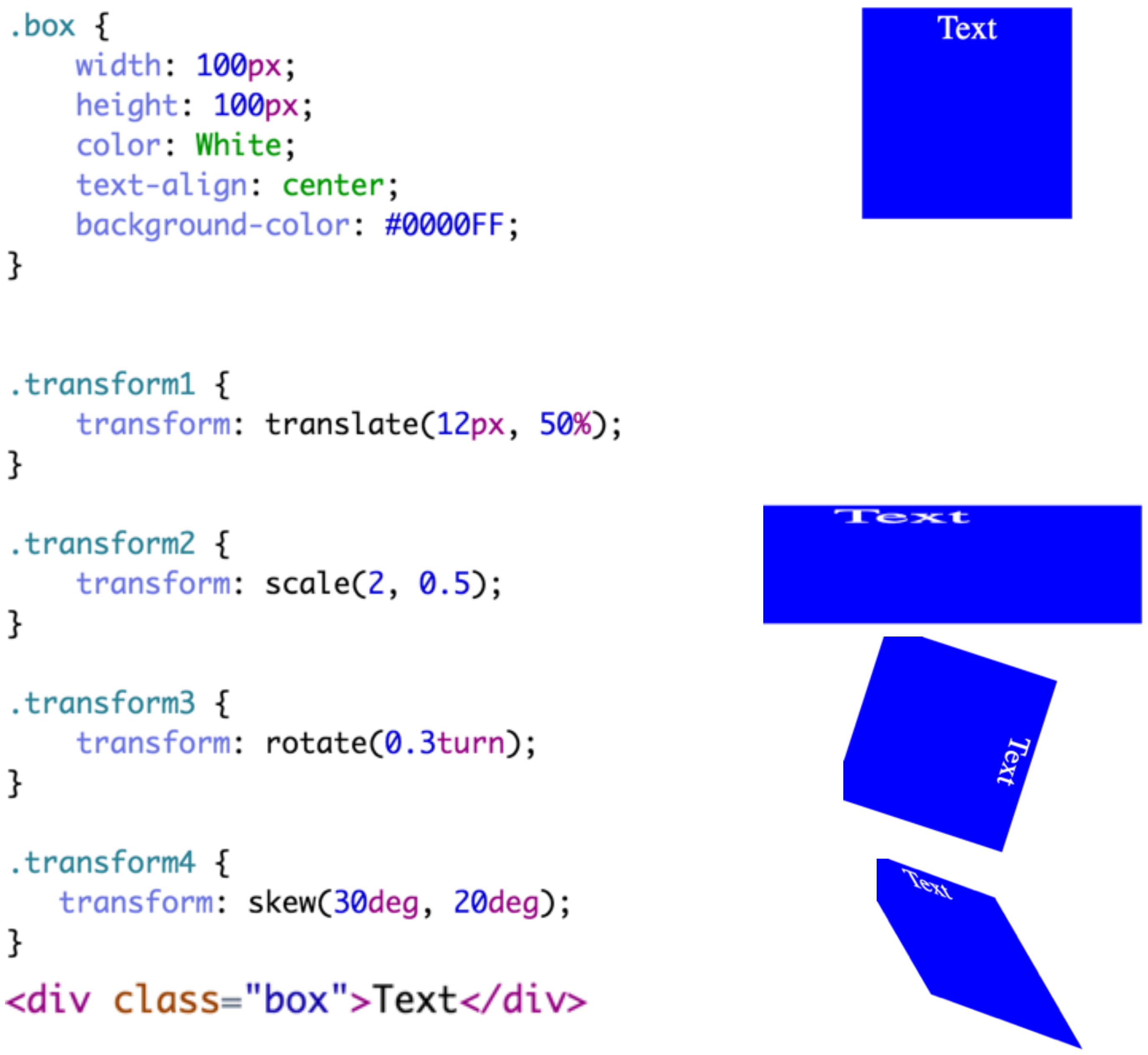
eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

- Elements taken out of normal flow may be stacked on top of each other
- Can set order with z-index property
  - Higher numbers appear in front
- Can set opacity of element, making occluded elements partially visible

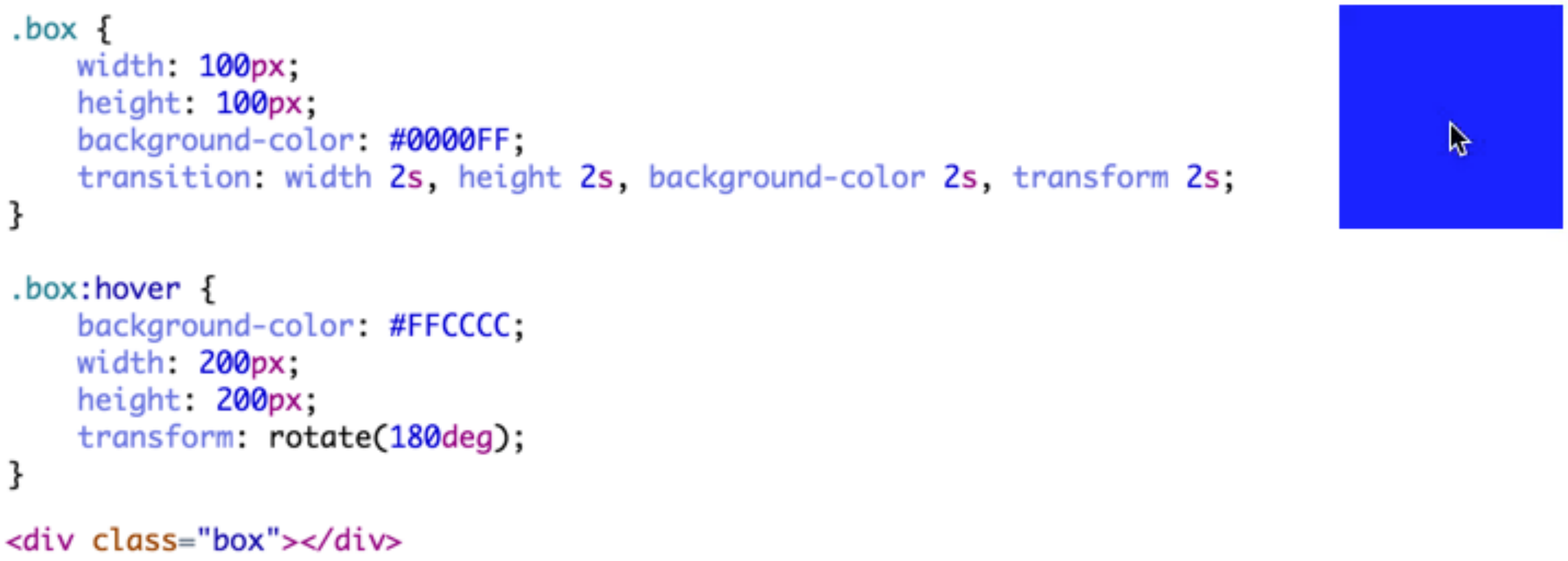
Transform - examples



```
.box {
    width: 100px;
    height: 100px;
    color: White;
    text-align: center;
    background-color: #0000FF;
}


.transform1 {
    transform: translate(12px, 50%);
}

.transform2 {
    transform: scale(2, 0.5);
}

.transform3 {
    transform: rotate(0.3turn);
}

.transform4 {
   transform: skew(30deg, 20deg);
}
<div class="box">Text</div>
```

- Can modify coordinate space of element to rotate, skew, distort

Transitions



```
.box {
    width: 100px;
    height: 100px;
    background-color: #0000FF;
    transition: width 2s, height 2s, background-color 2s, transform 2s;
}

.box:hover {
    background-color: #FFCCCC;
    width: 200px;
    height: 200px;
    transform: rotate(180deg);
}

<div class="box"></div>
```

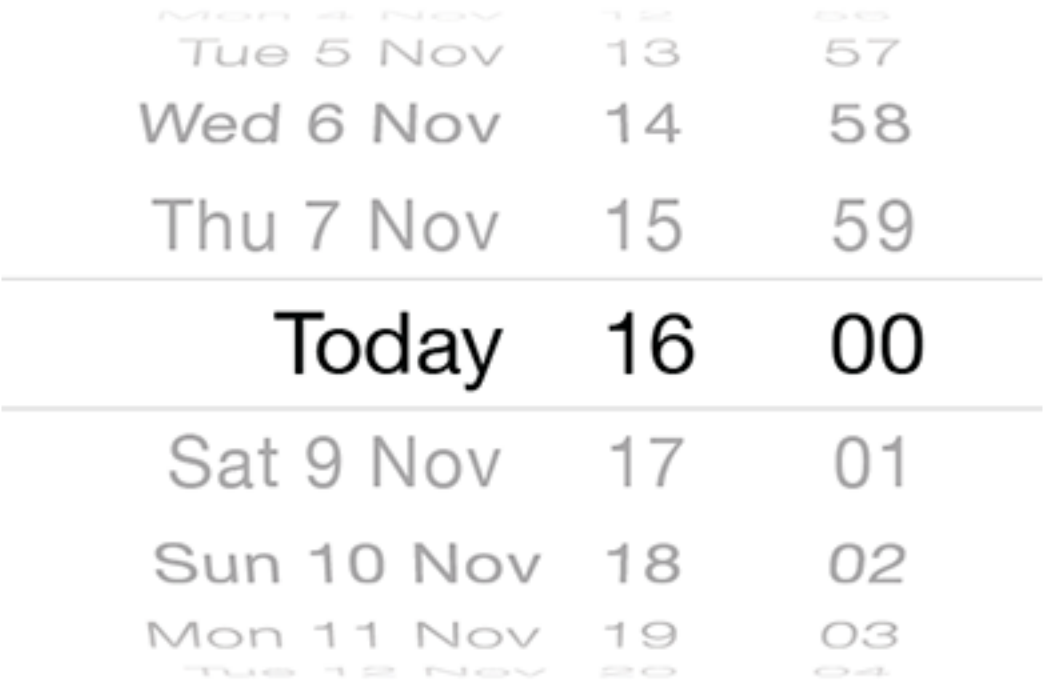
- transition: [property time], …, [property time]
  - When new class is applied, specifies the time it will take for each property to change
  - Can use *all* to select all changed properties

Fixed width vs. liquid layouts

- Fixed width
  - Use width="[num]px" to force specific sizes
  - Allows for tightest control of look and feel
  - But can end up with extra whitespace around edge of web page
- Liquid layout
  - Use width="[num]%" to size relative to container sizes
  - Pages expand to fill the entire container size
  - Problems
    - Wide windows may create long lines of text can be difficult to read
    - Very narrow windows may squash words, breaking text onto many lines
  - (Partial) solution
    - Can use min-width, min-height, max-width, max-height to set bounds on sizes

Designing for mobile devices

- Different devices have different aspect ratios.
  - Important to test for different device sizes.
  - May sometimes build alternative layouts for different device sizes.
- Using specialized controls important.
  - Enables mobile browsers to use custom device-specific widgets that may be much easier to use.

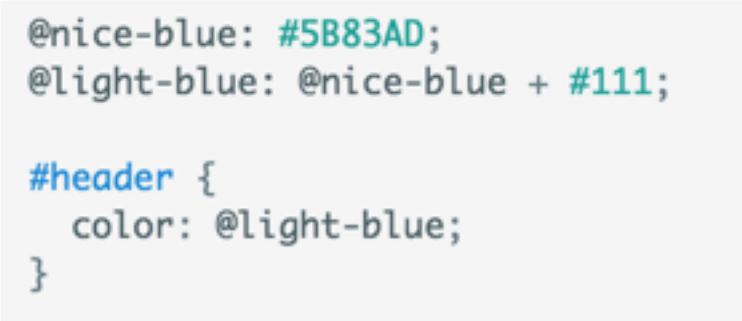


Mon 4 Nov 12 56
Tue 5 Nov 13 57
Wed 6 Nov 14 58
Thu 7 Nov 15 59
Today 16 00
Sat 9 Nov 17 01
Sun 10 Nov 18 02
Mon 11 Nov 19 03
Tue 12 Nov 20 04

CSS Preprocessors

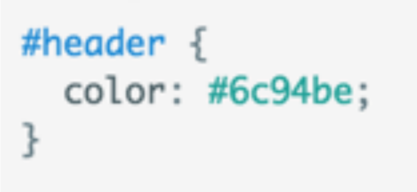
- Languages extend CSS to offer more powerful ways to specify rules (e.g., LESS, SASS)
  - Make rules more compact and less redundant
- Examples from LESS:

Variables



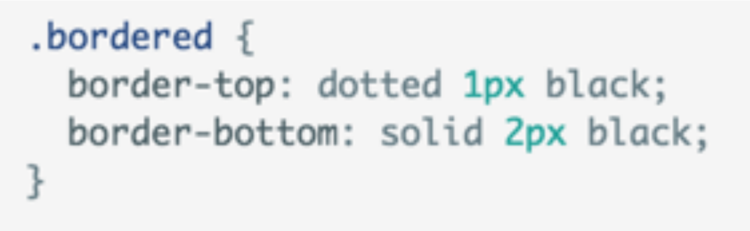
```
@nice-blue: #5B83AD;
@light-blue: @nice-blue + #111;

#header {
  color: @light-blue;
}
```

compiles to

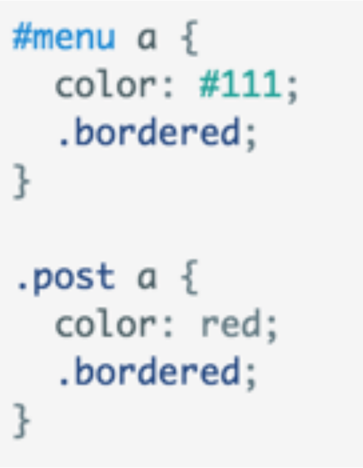


```
#header {
  color: #6c94be;
}
```

Mixins

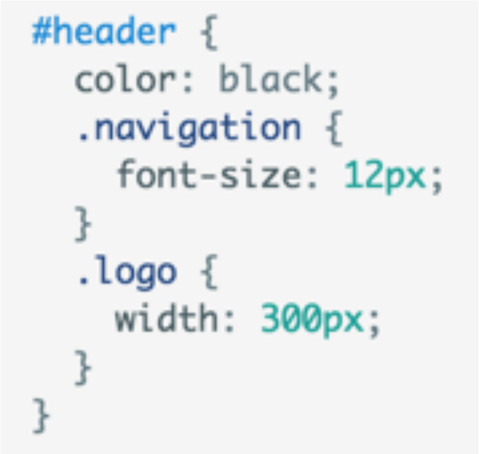


```
.bordered {
  border-top: dotted 1px black;
  border-bottom: solid 2px black;
}
```



```
#menu a {
  color: #111;
  .bordered;
}

.post a {
  color: red;
  .bordered;
}
```

Nested rules



```
#header {
  color: black;
  .navigation {
    font-size: 12px;
  }
  .logo {
    width: 300px;
  }
}
```

http://lesscss.org/features/

CSS Best Practices

- When possible, use CSS to declaratively describe behavior rather than code
  - Easier to read, can be optimized more effectively by browser
- Don’t repeat yourself (DRY)
  - Rather than duplicating rules, create selectors to style all related elements with single rule
- CSS should be readable
  - Use organization, indentation, meaningful identifiers, etc.

Supplementary Materials

- Tutorials, reference materials, and examples
  - HTML: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML
  - CSS: https://developer.mozilla.org/en-US/docs/Web/CSS
- Pastebin for experimenting with HTML & CSS
  - https://seecode.run

In Class Activity

Style the website that you made at the beginning.

You should try to use CSS features such as

- class selectors, id selectors, descendent selectors, etc.
- pseudo classes
- alternative positioning schemes