HyperText Markup Language (HTML5)

PIC 40A, UCLA

HTML

HTML (hypertext markup language) is a markup language (not a programming language) for how information on a webpage should be rendered/displayed in a web browser.

Its most modern incarnation is HTML5, which has extended the capabilities of previous versions by adding **semantics** to different elements of a webpage and allowing special functionalities that were previously only possible through more advanced scripting and styling.

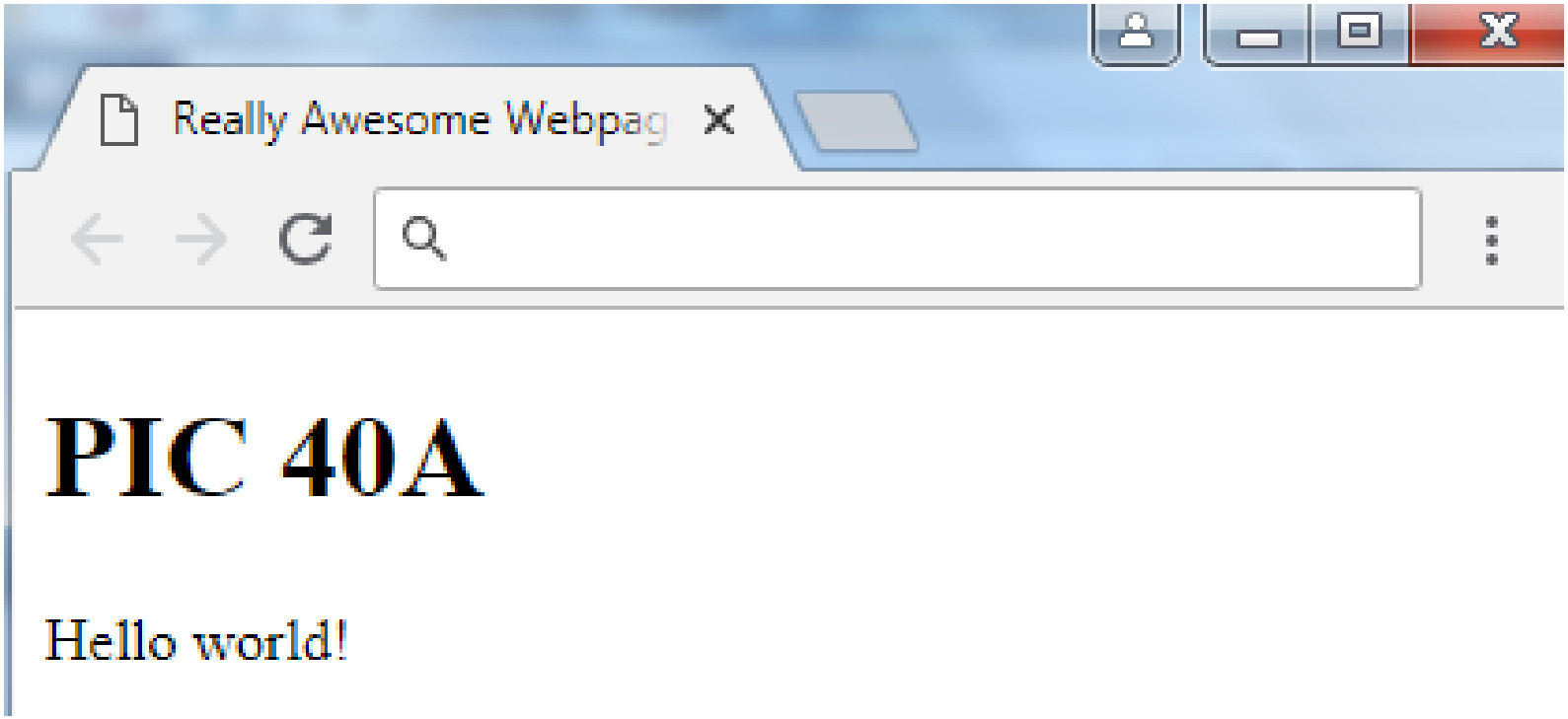
A Simple Webpage

Here is a very simple HTML5 webpage:

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <!-- some html -->
5  <head>
6    <meta charset="UTF-8">
7    <title>Really Awesome Webpage</title>
8  </head>
9
10 <body>
11   <h1>PIC 40A</h1>
12   <p>
13     Hello world!
14   </p>
15 </body>
16
17 </html>
```

A Simple Webpage

This renders the output



A Simple Webpage

Let's take this apart in stages...

<!DOCTYPE html>

This tells the web browser that the page is formatted according to HTML5 guidelines. If HTML4 were used instead, it would like like
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

A Simple Webpage

<!- - some html - ->

Comments in HTML are prefixed by **<!- -** and terminate with **- ->**. They can be multi-line.

A Simple Webpage

```
<html lang = "en">
   ...
</html>
```

HTML is built around **elements**. An element generally has a starting tag, some content, and an end tag, though not always. So **html** is a type of element; its starting tag is <**html lang = "en"**> and its ending tag is </**html**>. Its content is the **...** material.

A Simple Webpage

```
<html lang = "en">
   ...
</html>
```

A **tag** can have various forms:

- <**...**>: for a starting tag (or a tag that is empty)
- </**...**>: for an ending tag
- <**...**/>: for a tag that is empty.

A Simple Webpage

```
<html lang = "en">
   ...
</html>
```

An **attribute** gives various properties to an element. It appears within the start tag of the element.

Multiple attributes can be given as space separated **name = "value**”-pairs.

In the example above we have told the browser that the primary language of the webpage is English. The attribute is **lang** for language and its value is **"en"** for English.

A Simple Webpage

```
<head>
   ...
</head>

<body>
   ...
</body>
```

Within the HTML tags, there should be **head** and **body** elements. The **head** element stores meta-data about the page: its title, style information, keywords, etc.

The **body** element stores all the real content.

A Simple Webpage

Remark: the taggings are not case sensitive. We could just as well use **<HEAD> ... </hEaD>**, etc.

HTML can be quite forgiving syntactically. Missing a closing tag, having a tag that is not actually part of the **W3C Recommendations** (World Wide Web Consortium "rules" about HTML), could all yield webpages that work perfectly fine.

Despite its forgiving nature, web developers should strive to write markup that is easy to read and robust (there are more strict interpretations of HTML, XHTML, which would not allow such carelessness). Also to be robust against future updates to the standards, it is best to be rigid in format/style.

A Simple Webpage

```
<head>
   <meta charset = "UTF-8">
   <title>Really Awesome Webpage</title>
</head>
```

The content of the **title** element sets the page title that the web browser (should) display. The title element is required for the head of HTML documents!

The **meta** element is **void**, with its start being its end tag. Here it specifies the character set being used is UTF-8. By default this is what is used. This **charset** is a new attribute in HTML5.

A Simple Webpage

```
<body>
   <h1>PIC 40A</h1>
   <p>
      Hello world!
   </p>
</body>
```

Within the **body** element, we have a **heading** element, **h1**, and a **paragraph** element, **p**.

A Simple Webpage

<h1>PIC 40A</h1>

The **heading** elements are: **h1, h2, h3, h4, h5, h6** in order of largest size/greatest emphasis down. We might use **h1** for a title; **h2** for a subtitle; etc.

Remark: it is important to not mix up the heading level elements and the **header** element (HTML5).

A Simple Webpage

```
<p>
  Hello world!
</p>
```

The paragraph stores the message "Hello world!" to display. When a browser sees a paragraph, by default it will pad the text with a little bit of space, seeing it as a separate entity from the other elements.

A Simple Webpage

Remarks:

- Besides spaces, white space is not added when there is a line break in the element's content. To add line breaks, we use other elements such as **
**.
- Note the indentation: HTML documents can have tags within tags within tags and it can be kind of hard to read. Style guidelines will be discussed, but just be aware that it does help with readability.

Family Tree

In describing elements and their relationships, we use the terms **parent**, **child**, and **sibling**s.

The **body** element is a parent element to **h1** and **p**.

The **h1** and **p** elements are children of **body**.

The **h1** and **p** are siblings.

Semantics

Previous versions of HTML used elements for organizational and layout purposes...

HTML5 does all of this, but the elements also have a **semantic meaning**. Proper use of HTML5 means paying attention to the semantics. The semantics help web browsers in displaying information and making the website more accessible. Search engines can also more easily find the page content.

Some Basic Elements

We will go through a series of elements to indicate their purpose and semantics.

<header>...</header>: the **header** element represents material that should appear at the top of a webpage such as the title, subtitle, author, and maybe some navigation.

<main>...</main>: the **main** element represents the bulk of the page that is unique or special content, excluding the header or footer.

<footer>...</footer>: this is stuff that goes at the bottom of the page such as copyright statement, indemnity info, links to other sites, etc.

Some Basic Elements

h1-h6 heading elements: besides **h1** for title, and **h2** for a subtitle, etc., these should generally following the convention that $\mathbf{h}_{i+1}$ gives a section title, where that section is a subsection of another section titled by $\mathbf{h}_i$.

<small>...</small>: for a legal disclaimer, copyright, last updated info, etc.

Some Basic Elements

** ... **: paragraph elements are used for paragraphs but more generally as items/thoughts that are grouped thematically, e.g., the section of a web form for the spice level of an Indian curry.

<article>...</article>: an entity unto itself such as an entire blog post, etc.

Some Basic Elements

<section>...</section>: bigger than a paragraph, a collection of linked ideas: introduction, methods, conclusions for a scientific article, for example. Its proper use requires either an **aria-label** or **aria-labelledby** attribute.

Each element of a page can be given an **id** attribute to uniquely identify it. It must contain at least one character and have no spaces. Older HTML versions were more restrictive. The ID must be unique for each element.

The section should either have **aria-labelledby="id_of_heading"** (referring to the heading that labels that section) or **aria-label="label_of_section"**. Without this extra label, the semantics are lost and the website is less accessible.

aria=Accessible Rich Internet Applications are a set of attributes that help make websites more accessible to those with disabilities.

Some Basic Elements

In the past, the ** ... ** element, with no semantic meaning, was the only real choice for dividing a page into different parts. **Article** and **section** should often be used for HTML5 documents instead of **div**'s.

A **...** element also lacks semantics but can be useful in CSS for inline styling. By default **span**s are inline and **div**s stack.

Special Entities

Because **<** and **>** are used for tags, we need special escape sequences to represent them:

< renders **<**

> renders **>**

Then, because **&** is used in an escape sequence, we use:

& to render **&**. Another useful one is:

© to produce ©.

If you want two words to be spaced but not appear on separate lines, you can use a non-breaking space: ** **

Tag Semantics I

Consider the small excerpt:

```
1  <body>
2    <main>
3      <article>
4        <h1>PIC Classes</h1>
5        <section aria-labelledby="p10a">
6          <h2 id="p10a">PIC 10A</h2>
7          <p>
8            An introduction to C++...
9          </p>
10       </section>
11       <section aria-labelledby="p40a">
12         <h2 id="p40a">PIC 40A</h2>
13         <p>
14           Introduction to web programming...
15         </p>
16       </section>
17     </article>
```

Tag Semantics II

```
18    </main>
19    <footer>
20      <small>
21        We are not responsible for anything.
22      </small>
23    </footer>
24  </body>
```

Tag Semantics I

This renders the following output.

PIC Classes

PIC 10A

An introduction to C++...

PIC 40A

Introduction to web programming...

We are not responsible for anything.

The headings display the same size due to the default behavior of the web browser when elements are nested. This can be fixed with **CSS**.

Formatting Tags

Most formatting should be done with CSS, but before then we'll look at a few options for formatting.

...: *The b element represents a span of text to which attention is being drawn for utilitarian purposes without conveying any extra importance and with no implication of an alternate voice or mood, such as key words in a document abstract, product names in a review, actionable words in interactive text-driven software, or an article lede.* - W3C

...: *The strong element represents strong importance, seriousness, or urgency for its contents.* -W3C

Often both are rendered as **bold-faced**.

Formatting Tags

...: *the i element represents a span of text in an alternate voice or mood, or otherwise offset from the normal prose in a manner indicating a different quality of text, such as a taxonomic designation, a technical term, an idiomatic phrase from another language, transliteration, a thought, or a ship name in Western texts.* -W3C

...: *The em element represents stress emphasis of its contents.*
The level of stress that a particular piece of content has is given by its number of ancestor em elements.
The placement of stress emphasis changes the meaning of the sentence. The element thus forms an integral part of the content. The precise way in which stress is used in this way depends on the language.-W3C

Often both are rendered as *italicized*.

Formatting Tags

<u>...</u>: *The u element represents a span of text with an unarticulated, though explicitly rendered, non-textual annotation, such as labeling the text as being a proper name in Chinese text (a Chinese proper name mark), or labeling the text as being misspelt.* -W3C

Usually the content gets underlined.

<s>...</s>: indicates that content is no longer relevant/up-to-date, ~~strikes out the content~~.

~~cost $4.99~~
cost $3.39

Formatting Tags

**
**: renders a line break. Should only be used when a line break is part of the written content, not to add spacing between elements in general.

<hr>: represents a thematic break in content, often producing a horizontal rule (horizontal line dividing above/below).

_{...}: places content as a subscript.

^{...}: places content as a superscript.

<pre>...</pre>: content should be displayed exactly as it is with no special formatting. This can be useful in writing code, etc.

Anchors

The HyperText in HTML refers to being able to link to other pages and documents. We'll now look into linking!

The **anchor** element, **<a>...** is used for this purpose. We'll look at 3 of its attributes:

- **href**: specify a URL or relative path from the location of the current HTML file or an element's ID.
- **target**: options for values include **_blank** (open in new window), **_self** (open in same frame), **_parent** (open in parent frame), and **_top** (open in top of window).
- **title**: what is displayed when one mouses over the link.

It is possible to embed documents within other documents in HTML with frames.

Anchors

The link below would take someone to google.com to search. It would open up in a new tab/window. And the displayed text would be "Search Here".

```
<a href="http://www.google.com" target="_blank">Search Here</a>
```

The link below would take someone to the file **cat.png** stored one directory up and in the **pics** folder. It may/may not open up in the same window. When the cursor hovers over the link it would say "Cat Picture" and the displayed text would be "cute cat".

```
<p>
   A really <a href="../pics/cat.png" title="Cat Picture">cute cat</a>.
</p>
```

ID Attributes

An anchor can link to these elements with a given ID, taking someone to another part of the page.

For example, we could have a section of the webpage

```
<section id="current_projects" aria-labelledby="current">...</section>
```

And then provide a link there

```
<p>
   Check out the <a href="#current_projects">current projects</a> at our
   company.
</p>
```

iframe I

To embed one document within another, we can use **iframe**. The **iframe** element can have no content. It links to another document through the **src** attribute.

outer.html

```
1 <main>
2   <p>Top Outer</p>
3   <iframe src="inner.html"></iframe>
4   <p>Bottom Outer</p>
5 </main>
```

iframe II

inner.html

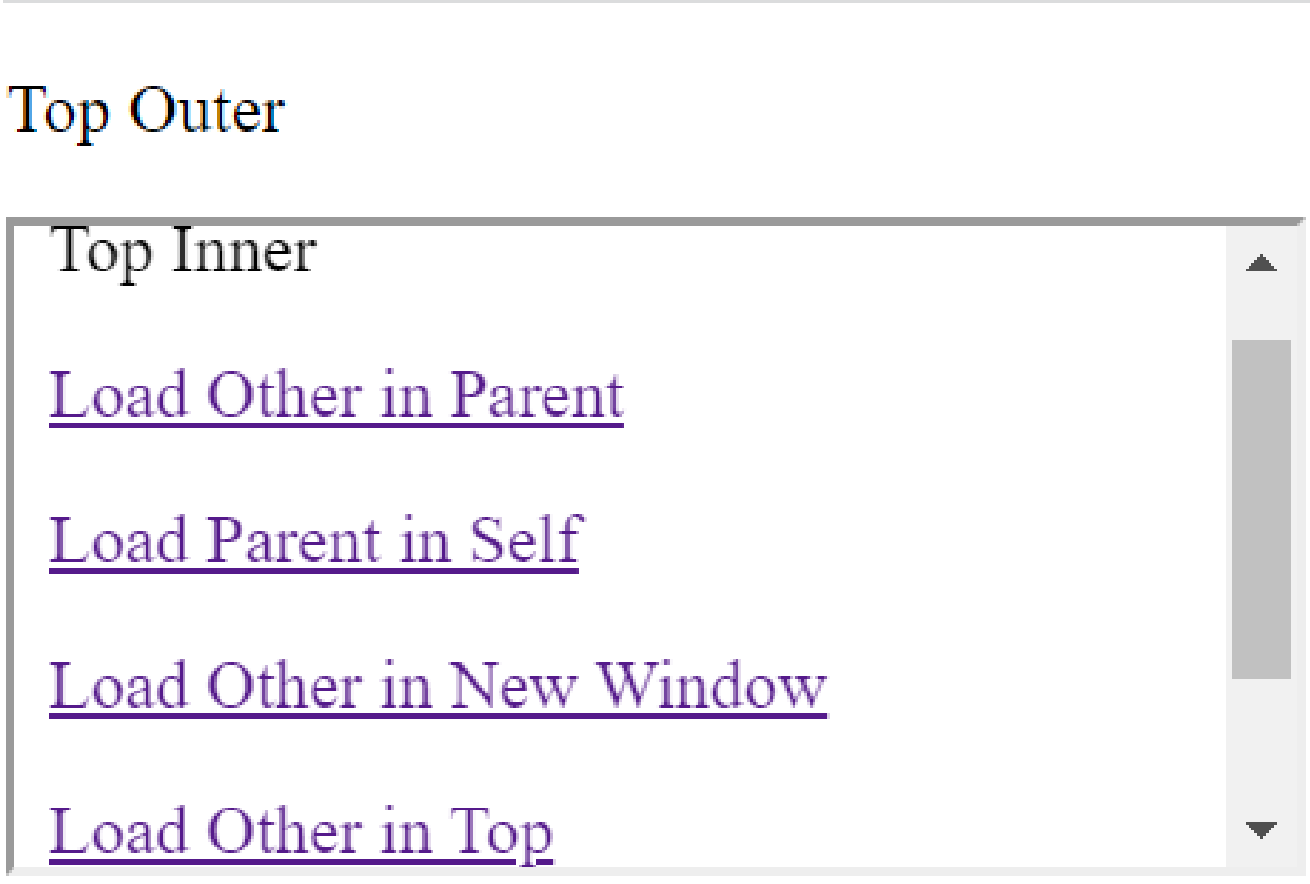
```
1  <main>
2    <p>Top Inner</p>
3  <p><a href="other.html" target="_parent">Load Other
     in Parent</a>
4  <p><a href="outer.html" target="_self">Load Parent in
     Self</a>
5  <p><a href="other.html" target="_blank">Load Other in
     New Window</a>
6  <p><a href="other.html" target="_top">Load Other in
     Top</a>
7  <p>Bottom Inner</p>
8  </main>
```

iframe III

other.html

```
<body>
  <main>
    <p>Other Page</p>
  </main>
</body>
```

iframe IV



Basic configuration

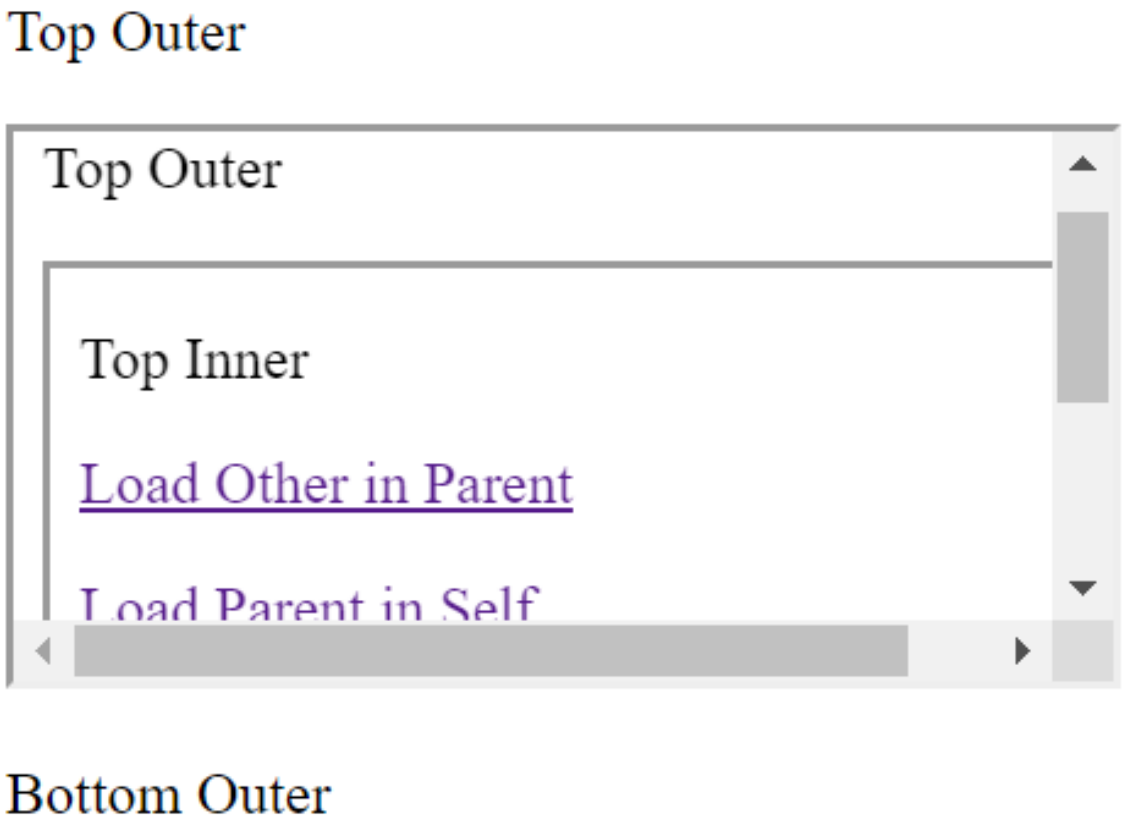
iframe V

Any loading of **other.html** with **_blank** will open **other.html** in a new window.

Any loading of **other.html** in **_top** will take the entire window in which the frame is found to **other.html**.

iframe VI

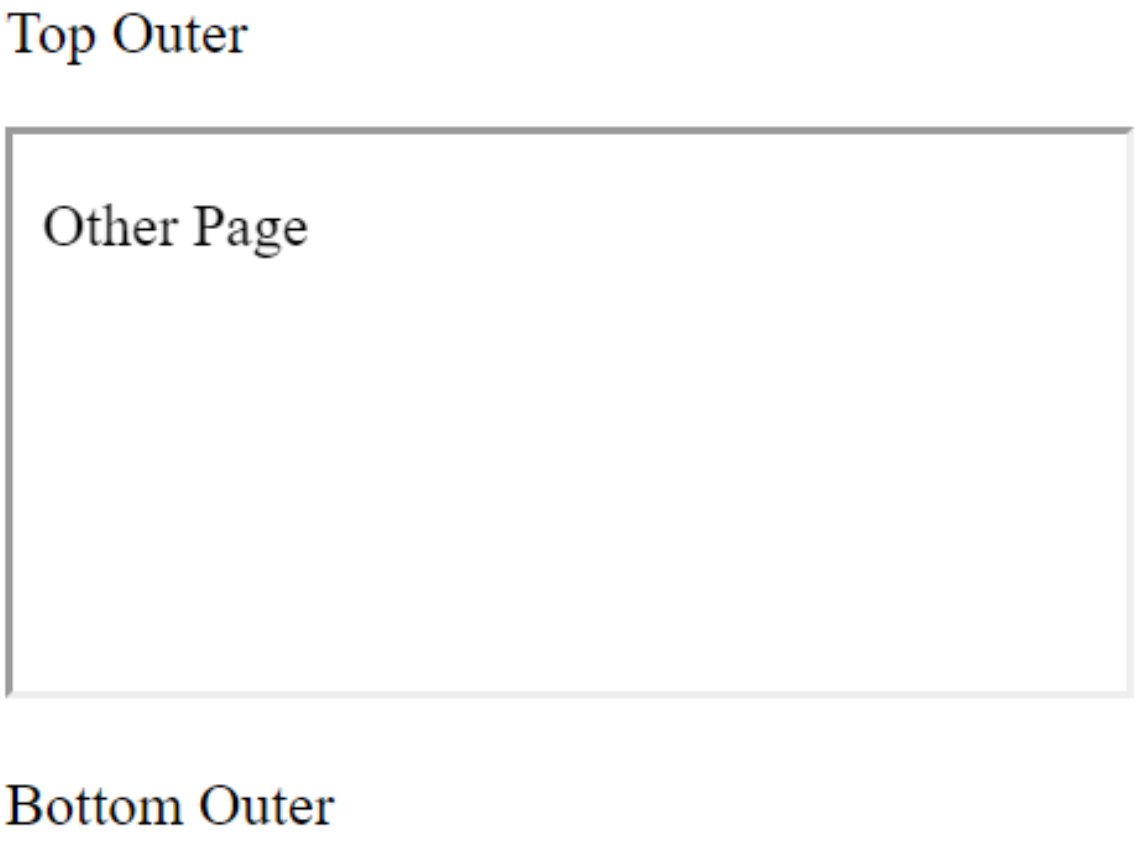
Amusingly, we can load the content of the parent frame, **outer.html** inside of **inner.html** by loading **outer.html** with **_self**.



Loading parent in self

iframe VII

And from any frame in which **other.html** is loaded with **_parent**, the parent frame will be redirected.



Loading other in parent

Lists

HTML supports 3 types of lists: **unordered lists**, **ordered lists**, and **description lists**.

An unordered list **
...
** is used when the order is unimportant, such as a bullet list of points in no particular order.

An ordered list **
...
** is used when the items should be ordered in sequence. A **type** attribute can be set with values "1", "A", "a", "i", "I" for (1,2,3,4,...)-, (A,B,C,D,...)-, (a,b,c,d,...)-, (i, ii, iii, iv, ...)-, (I, II, III, IV, ...)-indexing. The default is "1".

All items in an ordered/unordered list appear within a **list item** element, **
...
**.

Lists

A description list **<dl>...</dl>** is used to link items with their descriptions.

Within a definition list, we have the

- **description term** element, **<dt>...</dt>** and the
- **description details** element, **<dd>...</dd>** to describe the term.

The list used to be called a **definition list** but in HTML5 it is a **description list**.

Lists I

To generate a page such as the one below

Yummy Smoothie Ingredients

- frozen cherries or frozen banana
- avocado
- cacao powder
- almond milk
- dates

Instructions

1. place ingredients inside a blender
2. blend
3. enjoy

About the Ingredients

avocado
- a green and delicious fruit with a creamy texture, makes the drink more like a rich pudding

almond milk
- a healthy alternative to milk, but should be unsweetened or the recipe may be too sweet

dates
- a healthy alternative to refined sugar that does not lack in sweetness!

Lists II

we use code:

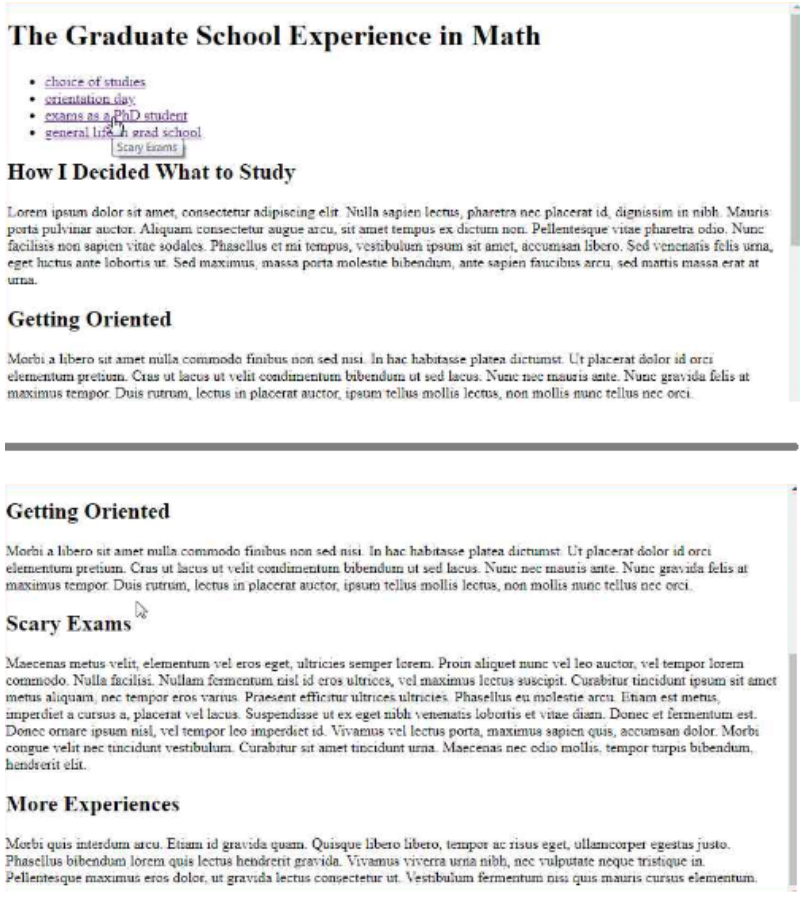
```
<h2>Yummy Smoothie Ingredients</h2>
<ul>
    <li>frozen cherries or frozen banana</li>
    <li>avocado</li>
    <li>cacao powder</li>
    <li>almond milk</li>
    <li>dates</li>
</ul>

<h2>Instructions</h2>
<ol>
    <li>place ingredients inside a blender</li>
    <li>blend</li>
    <li>enjoy</li>
</ol>

<h2>About the Ingredients</h2>
```

Lists III

```
18 <dl>
19   <dt>avocado</dt>
20   <dd>
21       - a green and delicious fruit with a creamy
           texture, makes the drink more like a rich
           pudding
22   </dd>
23   <dt>almond milk</dt>
24   <dd>
25      - a healthy alternative to milk, but should be
          unsweetened or the recipe may be too sweet
26   </dd>
27   <dt>dates</dt>
28   <dd>
29       - a healthy alternative to refined sugar that
           does not lack in sweetness!
30   </dd>
31 </dl>
```

Page Navigation

We'll now consider how to navigate a simple blog page such as:



Page Navigation I

Some of the relevant code is included.

```
1  <body>
2    <header>
3      <h1>The Graduate School Experience in Math</h1>
4      <nav>
5        <ul>
6          <li><a href="#decision" title="How I Decided
               What to Study">choice of studies</a> </li>
7          <li><a href="#orientation" title="Getting
               Oriented">orientation day</a> </li>
8          <li><a href="#quals" title="Scary Exams">exams
               as a PhD student</a> </li>
9          <li><a href="#more" title="More Experiences"
               >general life in grad school</a> </li>
10       </ul>
11     </nav>
12   </header>
13
```

Page Navigation II

```
14    <main>
15      <article id="decision">
16        <h2>How I Decided What to Study</h2>
17        <p>Lorem ipsum dolor sit amet, consectetur
              adipiscing elit. Nulla sapien lectus, pharetra
               nec placerat id, dignissim in nibh. Mauris
              porta pulvinar auctor. Aliquam consectetur
              augue arcu, sit amet tempus ex dictum non.
              Pellentesque vitae pharetra odio. Nunc
              facilisis non sapien vitae sodales. Phasellus
              et mi tempus, vestibulum ipsum sit amet,
              accumsan libero. Sed venenatis felis urna,
              eget luctus ante lobortis ut. Sed maximus,
              massa porta molestie bibendum, ante sapien
              faucibus arcu, sed mattis massa erat at urna.
              </p>
18      </article>
```

Nav Element

A **nav** element is used to hold navigational information about a webpage. Often they appear in the **header** element and provide links to different parts of the webpage or another page on the website.

Think of a **<nav>...</nav>** like giving a table of conents with links to other part of the page.

Nav Element

Aside: the text in the blog is called Lorem Ipsum text. It is designed to have similar distributions of letters and word length as English but is mostly gibberish. Web developers use this text when they are focused on getting a page layout and don’t want to be distracted by the content.

HTML Style Guidelines

It is important to come up with some guidelines for writing good HTML code. These are important to ensure other people can read your code and that it is robust to possible changes in how web browsers parse HTML, etc.

HTML Style Guidelines

- Always declare the doctype **<!DOCTYPE html>**.
- Always include the **html**, **head**, and **body** elements.
- Always indent child elements unless they are the **head** or **body** elements. The indenting could be a tab or 4 spaces.
- Always use lowercase for the element names for consistency and readability.
- Use proper semantics.
- Use **aria-labelledby** or **aria-label** with the **sections**.
- Validate your code with an HTML5 validator such as: **this one** - do not have misaligned or missing opening/closing tags.

Search Engine Optimization

Just because a website is live on the web does not mean it is easy for people to find it through a search engine. Around 1997, web developers took an active interest in this problem and gradually more research went into how to show up in search results.

This has led to a field of online marketing known as **Search Engine Optimization (SEO)**. It assimilates strategies that help a webpage to rank more highly in search results.

Search Engine Optimization

There are two main SEO practices: **white hat** (producing high quality content that users genuinely benefit from, while making content easier to find for search engines) and **black hat** (using deceit, plagiarizing content, and rigging things to be found by search engines but offering little of value).

Search engines these days are quite smart and they tend to penalize black hat practices and give higher ranks to relevant, white hat sites.

Search Engine Optimization

We consider a few practices of the white hat SEO strategies, but really an entire course could be given on SEO alone.

- Choose an appropriate page title!
- Use the **meta** tag with the **name** property as in:
  <meta name="keywords" content="homelessness, point-in-time counts, Los Angeles">
- Ensure the website is easy to navigate with a logical structure in how the pages are put together.
- Use HTML5 semantics well!

Search Engine Optimization

- Research **keywords** that are relevant to your site and incorporate them in headings and the text content with high frequency, but *organically*. For example if you run meditation classes, having a page section titled "try this" would be less effective than "Guided meditation" because many may search the string "guided meditation". **Click here for a useful keyword tool.**
- Get linked to by other reputable websites (and link to others).

Meta

A given document can have multiple **meta tags**. In general, the **content** attribute sets the value for either the named metadata (specified through **name** like “keywords") or the **http-equiv** setting (to help control caching of the page, etc.).

Some other named metadata include things like **author** and **description**.

Meta

We can prevent browser caching (could prevent visitors from seeing the new page if HTML changes) with:

```
<meta http-equiv="Cache-Control" content="no-cache, no-store, must-revalidate" />
<meta http-equiv="Pragma" content="no-cache" />
<meta http-equiv="Expires" content="0" />
```

Tables

The **table** element, **<table>...</table>** allows us to write tables in HTML.

A table can have a

- **table head**, **<thead>...</thead>** for giving the names of column fields, etc.;
- **table body**, **<tbody>...</tbody>** for the data; and
- **table footer**, **<tfoot>...</tfoot>** for other analysis such as giving the sums of the columns, average, etc.

Tables

The **table header** element, **<th>...</th>** signifies that something is a header or label.

This can apply to row labels, too, not just column labels.

Each row is represented by a **table row**, **<tr>...</tr>** element and every piece of data is represented by a **table data** element, **<td>...</td>**.

The **th** and **td** elements can be empty, too, so there is an empty string inserted.

Tables I

To get a display such as

Student	Midterm	Final
#1	80	
#2	70	60
#3	30	30
Mean	60	45

Tables II

we write the code

```
<table>
  <thead>
    <tr>
      <th>Student</th> <th>Midterm</th> <th>Final</th>
    </tr>
  </thead>

  <tbody>
    <tr>
      <th>#1</th> <td>80</td> <td><!--BLANK--> </td>
    </tr>
    <tr>
      <th>#2</th> <td>70</td> <td>60</td>
    </tr>
    <tr>
      <th>#3</th> <td>30</td> <td>30</td>
    </tr>
```

Tables III

```
18    </tbody>
19
20    <tfoot>
21      <tr>
22        <th>Mean</th> <td>60</td> <td>45</td>
23      </tr>
24    </tfoot>
25  </table>
```

We could just as well have replaced **<td><!–BLANK–></td>** with **<td></td>** - recall that comments are not read.

Remark: note we have to "hard code" the values into the table. Since HTML is not a language, there isn't really another way. But with JavaScript and PHP, this can be automated.

Images

The **image** element is self-closing, **<img ... >** or ****. Within the element, we can place some attributes and values:

- **src**, the source of the image - should be a URL or relative path;
- **alt**, the alternate text to display should the image fail to load, web browsers also use this for the visually impaired
- **height**, the height in pixels (px); and
- **width**, the width in pixels.

If only one of height/width is specified, the aspect ratio will be preserved. Otherwise we may distort the image.

Images

Two related semantic elements are the **figure** and **figcaption** elements.

We use **figure** to indicate that an image is a figure, and its caption can appear within the **figcaption** element. The figure can stand alone, outside of the text.

Images

An example of including a photo of Chichen Itza is below:

```
<p>
  The Mayans had a highly sophisticated knowledge of
     astronomical events. One particularly striking
     example is Chichen Itza, one of the Mayan
     pyramids.
</p>
<figure>
  <img src="ChichenItzaEquinox.jpg" alt="View of
     descending serpent of light formed along Chichen
     Itza pyramid" title="Chichen Itza at the Equinox"
     width="300">
  <figcaption>
    On the autumn equinox, a descending serpent of
       light forms along the steps of Chichen Itza.
  </figcaption>
</figure>
```

Images

The Mayans had a highly sophisticated knowledge of astronomical events. One particularly striking example is Chichen Itza, one of the Mayan pyramids.

On the autumn equinox, a descending serpent of light forms along the steps of Chichen Itza.

Audio

The **audio** element allows for audio files to be played, **<audio>...</audio>**.

It should always have a **controls** attribute, assigned no value, to ensure controls are displayed.

Within the element, we can provide a series of audio files for the browser to try playing; it will try the files in order until the first file it can play. If will not play anything if none of the files can be played.

A text message can be displayed in the case the browser does not support the element.

To list the audio files, we use the **source** element with a single **src** attribute.

Audio

```
<p>
  Birds are nice. Let's listen to some birds.
</p>
<audio controls >
  <source src="bird.mp3"/>
  <source src="bird.wav"/>
  You are out of luck. Your browser does not support
     this element.
</audio>
```

Birds are nice. Let's listen to some birds.

Audio

Remark: the **controls** attribute is a **boolean attribute**. This is why it is not set to anything. Its presence is what ensures a certain functionality is present, the presence of play/pause buttons, etc.

Video

The **video** element behaves similarly to the audio element. It should have a **controls** attribute with no value set and it can contain multiple video files to play or an error message if the element is not supported. It is also possible to specify a height and/or width.

```
1 <video width="320" controls >
2   <source src="bird.mp4"/>
3   <source src="bird.ogg"/>
4   You are out of luck. Your browser does not support
       the HTML5 video element.
5 </video>
```

Anchor Security

When page visitors are redirected to another site through an anchor, some care is needed to ensure the **rel** property is set to **"noopener"**. This is the default these days but page visitors could try to write their own hyperlinks where the **rel="opener"**. This can lead to security issues.

When we study JavaScript, we will see how this can be a danger. **Read more about it here!**

Combining Anchors and Other Elements

We can turn almost anything into a link. Here are two examples:

```
<h3><a href="www.somepage.com" target="_blank"
rel="noopener">Some Page</a></h3>

<a href="www.anotherpage.com" target="_blank" rel="noopener"><img
src="cat.png" alt="adorable cat" /></a>
```

In the first case, the **h3** heading, when clicked, takes the user to **somepage.com**. In the second case, when the image is clicked, it takes the user to **anotherpage.com**.

Web Forms

Of course the internet these days would not be complete without being able to log in to social media websites, do online shopping, etc. To that end, we must study HTML web forms, the forms that accept user information: name, password, multiple choice selections, special instructions, etc.

Let's look at a simple form to get a sense of the structure.

Web Forms

```
1  <form>
2    <p>
3      <label for="littleTextArea">Enter something here:
           </label>
4      <input type="text" name="theirMessage" id="
           littleTextArea" />
5    </p>
6    <p>
7      <input type="reset" value="Clear All." />
8      <input type="submit" value="Click me." />
9    </p>
10 </form>
```

Web Forms


Enter something here:
Clear All
Click me.
Enter something here:
hello!
Clear All
Click me.
Enter something here:
Clear All
Click me.

Web Forms

Content of a web form is contained within the **<form>...</form>** tags.

The **label** element specifies an attribute **for**, referring to the ID of an input element. Its content stores the text for that element.

Stictly speaking it is not needed, but by having a label element, the box/radio button will be toggled appropriately or the textbox selected should the user click on the text within the label element. For example, if the user clicked anywhere on "Enter something here:", whatever they typed next would be input for the text field, even though they didn't click inside the text field.

Web Forms

There are many types of **input** elements, depending on their **type** attribute. The "text" type means we want a small area to enter text.

The **name** attribute is useful when we process data from forms. Unless the input has a name, it is not collected!

Web Forms

In terms of buttons, when the **<input type="submit"...** /> we have a submit button. This submit button is given the text based on the **value** attribute.

Currently the form does nothing when submitted. Later on, we can use JavaScript and PHP to do more.

When the **<input type="reset"...** /> is used, the button serves to clear all the data from the form. As with a submit button, the **value** attribute can set the button's text.

Web Forms I

Here's another example:

```
1  <form action="some_php_script_to_call.php">
2    <fieldset id="starWarsMovies">
3      <p>
4        Which Star Wars movie is the best?
5      </p>
6      <p>
7        <label for="pt1">Part I</label>
8        <input type="radio" id="pt1" value="part1" name="
             SWMovie" checked />
9        <label for="pt2">Part II</label>
10       <input type="radio" id="pt2" value="part2" name="
             SWMovie" />
11       <label for="pt3">Part III</label>
12       <input type="radio" id="pt3" value="part3" name="
             SWMovie" />
13       <label for="pt4">Part IV</label>
```

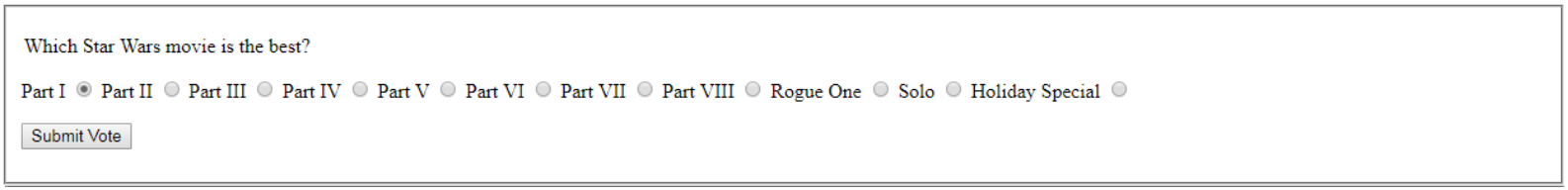
Web Forms II

```
14          <input type="radio" id="pt4" value="part4" name="
                SWMovie" />
15          <label for="pt5">Part V</label>
16          <input type="radio" id="pt5" value="part5" name="
                SWMovie" />
17          <label for="pt6">Part VI</label>
18          <input type="radio" id="pt6" value="part6" name="
                SWMovie" />
19          <label for="pt7">Part VII</label>
20          <input type="radio" id="pt7" value="part7" name="
                SWMovie" />
21          <label for="pt8">Part VIII</label>
22          <input type="radio" id="pt8" value="part8" name="
                SWMovie" />
23          <label for="r1">Rogue One</label>
24          <input type="radio" id="r1" value="rogue" name="
                SWMovie" />
25          <label for="solo">Solo</label>
```

Web Forms III

```
26          <input type="radio" id="solo" value="solo" name="
                SWMovie" />
27          <label for="special">Holiday Special</label>
28          <input type="radio" id="special" value="special"
                name="SWMovie" />
29        </p>
30        <p>
31          <input type="submit" value="Submit Vote" onclick=
                "" />
32        </p>
33      </fieldset>
34
35  </form>
```

Web Forms


Which Star Wars movie is the best?
Part I
Part II
Part III
Part IV
Part V
Part VI
Part VII
Part VIII
Rogue One
Solo
Holiday Special
Submit Vote

Web Forms

Related input elements can be grouped together within a **fieldset** element. Web browsers may put a box around the elements.

A radio button and check box are specified with the **<input type="radio"...**/> and **<input type="checkbox"...**/>.

They each receive a corresponding **label** and can be given a boolean attribute **checked** as to whether they are selected by default.

Web Forms

The **name** attribute for each grouping of buttons should be the same!

When dealing with inputs where multiple answers can be selected, in order to work with PHP scripts, the name will be of the form **name="something[]"** where the [] indicates that an array of data is collected.

Web Forms

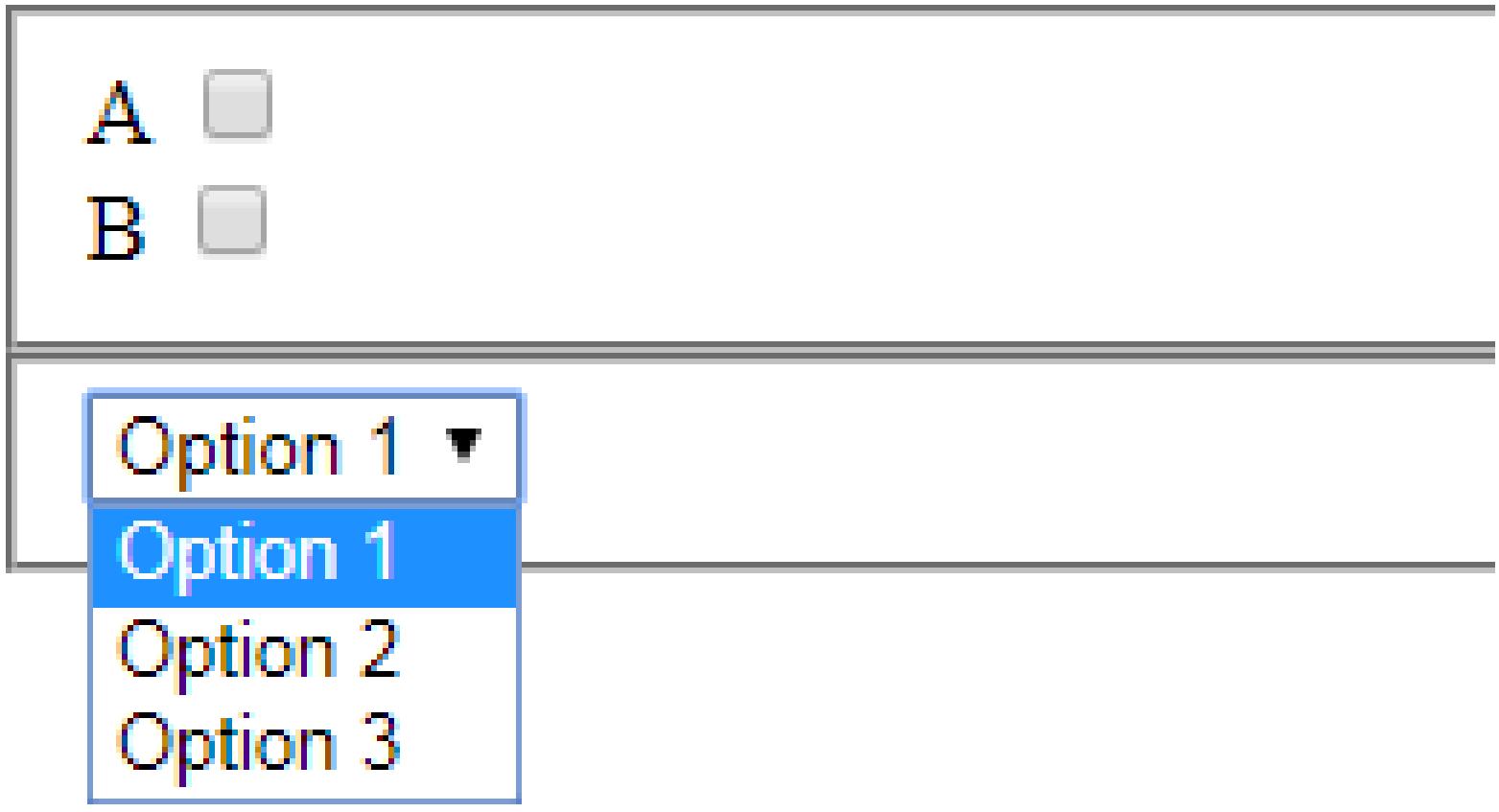
The **select** element, **<select>...</select>** gives a dropdown menu box to select an item.

The options are listed within the **option** elements, **<option value="someValue">...</option>** with a specified value.

Web Forms I

```
1  <form>
2    <fieldset>
3      <label for="A">A</label>
4      <input type="checkbox" id="A" value="A" name="
           options[]" />
5      <br>
6      <label for="B">B</label>
7      <input type="checkbox" id="B" value="B" name="
           options[]" />
8    </fieldset>
9    <fieldset>
10     <select name="OptionSelection">
11       <option value="Option1">Option 1</option>
12       <option value="Option2">Option 2</option>
13       <option value="Option3">Option 3</option>
14     </select>
15   </fieldset>
16 </form>
```

Web Forms



Web Forms

We'll just look at a few more form elements...
The **textarea** element, encompasses a large text area. In general, we would write:

```
<textarea name="whatever">Default Text if wanted</textarea>
```

Additional attributes of **rows** and **cols** can be given for the height and width of the textarea.

Web Forms

We can specify that we are accepting a password. Then what the user enters in the input is replaced by • or something else that obscures the real characters.

```
<input type="password" id="pass" name="pass" />
```

There are also inputs with **type="tel"** and **type="url"** for telephone numbers and URLs. In the latter case, the input can be validated before the form is submitted...

Web Forms

We can use an input element with **type="file"** to accept file submissions.

For **input** accepting files, emails, and the **select** element, we can specify a boolean attribute **multiple** allowing for multiple selections/submissions.

Web Forms

We can also sup up any input element by specifying a **pattern** attribute. The pattern attribute specifies a **regular expression** for what the input format should be.

A **regular expression** is a way of describing the set of all possible strings in a certain set (for example, the set of valid zip codes).

We could have an input that only accepts 5-digit zip codes with

```
<input type="text" pattern="[0-9]{5}" />
```

or one that only accepts phone numbers of the form **1-xxx-xxx-xxxx** or **xxx-xxx-xxxx** where each **x** is a digit from 0-9 with

```
<input type="tel" name="telnum" pattern="(1-)?[0-9]{3}-[0-9]{3}-[0-9]{4}"
   />
```

Web Forms

Here is some basic regex syntax. There is more syntax out there, but this is enough for most web forms.

- **[]**: we group a set of characters together with square brackets. [0-9] means the set of characters 0, 1, 2, ..., 9.
- **-**: used to indicate a range from character to its left to character to its right inclusive.
- **()**: parentheses are used for order of operations grouping characters together into a single element;
- **?**: zero or one instances of the preceding element;
- **{n}**: precisely n instances of the preceding element.

So **[0-9]{5}** indicates there must be precisely 5 characters from the set 0-9.

Many regex interpreters also interpret **d** as a digit, so we could have just as well written: **d{5}**.

Web Forms

- *****: zero or more instances of the preceding element;
- **+**: one or more instances of the preceding element;
- **.**: a wildcard, can be anything;
- **{n,}**: n or more instances of the preceding element;
- **{m,n}**: m to n inclusive instances of the preceding element.

We break **(1-)?[0-9]{3}-[0-9]{3}-[0-9]{4}** down into:

Zero or one copies of **1-**. Then contains 3 digits, a dash, 3 digits, a dash, and finally 4 digits.

Web Forms

Remarks:
If we actually mean the character **.**, we use **.**
Likewise to represent **?**, we use **?**, etc.
Within square brackets forming a set of characters, to represent **-**, we use **-**, etc.
To represent \, we use \ \ since \ is an escape character.

This is a useful Regex tester/debugger.

Web Forms

There are a few other Regex tricks that sometimes arise:

- **^**: starts with;
- **$**: ends with;
- **^**: also means excluding when in a set.

For **^cat[A-C]+\d+**, **catB444** passes the test.

For **[^\d]*\.txt$**, **myfile.txt** passes the test but **foo123.txt** does not (contains a digit).

Web Forms

For just one more example, an alphanumeric string of characters (digits 0-9, A-Z, a-z) can be described by:

[A-Za-z\d]+

Web Forms

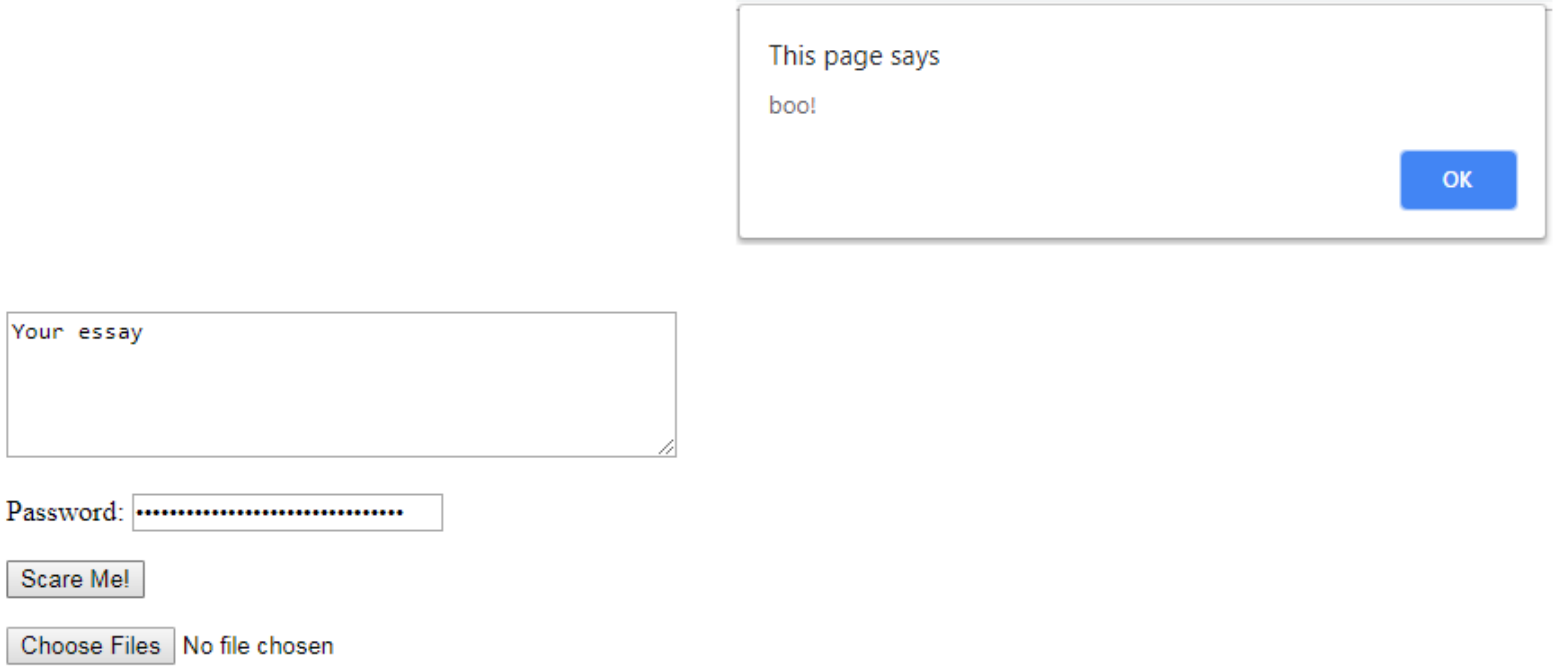
One can also create a generic button, not just submit/reset. Then through various event listeners, different scripts/code could be run when clicked. As with the submit/reset, the value specifies what the button displays.

```
<input type="button" value="Scare Me" />
```

In the above example, we suppose that the button has been configured through JavaScript to display an alert of “boo!" when clicked.

Web Forms

A textarea, password input, generic button, and file input are illustrated below:



Web Forms

A bit of relevant code:

```
1  <form>
2    <p>
3      <textarea rows="5" cols="50" name="essay">Your
           essay</textarea>
4    </p>
5    <p>
6      <label for="pass">Password: </label>
7      <input type="password" name="pass" id="pass" />
8    </p>
9    <p>
10     <input type="button" value="Scare Me!" />
11   </p>
12   <p>
13     <input type="file" name="files[]" multiple />
14   </p>
15 </form>
```

Web Forms

Remark: HTML does have a **<button>...</button>** element. There are varying opinions over which is better. The **button** element can turn an image into a button, say (by putting an **img** within the button). And **button**s are more conducive to styling and can work independent of a web form. The complaints primarily centre around some inconsistencies across browsers (cases are less and less frequent) and its default setting to submit a form when clicked (which can be changed).

Web Forms

We have so far seen **checked**, **controls**, and **multiple** boolean attributes. We will look at one more: **required**.

For text and similar inputs, this requires something be given in order for the form to be submitted. Unfortunately, HTML5 does not support this for check boxes, radio buttons, etc.

Block vs Inline

By default, each HTML element is either **block-level** or **inline**. Block elements stack (like paragraphs) and inline elements fit within blocks such as **b** or **a** elements.

Based on the elements we studied, here is a reasonably complete list of the block elements: **article, dd, div, dl, dt, fieldset, figcaption, figure, footer, form, h1, h2, h3, h4, h5, h6, header, hgroup, hr, li, main, nav, ol, p, section, table, ul**.

And a reasonably complete list of inline elements: **a, audio, b, br, em, i, input, img, label, s, select, small, span, strong, sub, sup, textarea, u, video**.

Remark: there are some annoying rules here. One states that you cannot nest block elements inside of a **p** element.

HTML5 Standards

The current version of the HTML5 Recommendations can be found **here** at the World Wide Web Consortium Website.

Summary

- HTML5 is the newest standardization of HTML and makes use of tag semantics.
- Tags have **property=“value”** pairings.
- Pages have **head**s for meta information and **body**s for content.
- For page sectioning, **p**, **section**, and **article** elements are used.
- Although often rendered the same, **i** and **em**, and **b** and **strong** have differing semantics.
- The **a** elements serve as anchors to link around the page and to other pages.
- **target** properties can have values **_blank**, **_self**, **_parent**, and **_top**.
- Documents can be embedded within documents with **iframes**.

Summary

- There are three types of lists: **ul**, **ol**, and **dl**.
- **SEO** is a strategy to drive more traffic to a website through search engines.
- For multimedia and graphics, HTML supports **img**, **audio** and **video**.
- Managing user input data can be handled on the front end through HTML **form**s.
- The **input**s support a **pattern** attribute for a regular expression.

Exercises I

1. Which part of an HTML document should titles, subtitles, etc. typically appear?
2. What is the purpose of the 'alt' attribute?
3. Give an example of where i is appropriate and where em is appropriate.
4. Write an HTML page for healthy lunches. Include an introduction to the page (write this for real - pay attention to semantics), two recipes (list real ingredients but feel free to use Lorem Ipsum for the instructions), and a web form where a user can sign up to the newsletter. The web form should ask for their name and email address. Both should be required fields and we define a valid email address as: *any nonempty string of characters followed by an '@' followed by any nonempty string of characters followed by a '.' followed by any nonempty string of characters.*
5. Explain **aria-label** and **aria-labelledby**.

Exercises II

6. Write a collection of webpages **index.html**, **A.html** and **B.html** where the main page has an **iframe** that can load one of two pages **A.html** or **B.html** in the frame, depending on the anchor clicked.