# CSCB20 – Week 6

*Introduction to Database and Web Application Programming*

*Anna Bretscher*
Winter 2016

## HyperText Markup Language

HTML documents may be:

- static, stored files returned by a Web server
- dynamically-generated by server-side script, such as PHP making SQL queries on a database to generate HTML at runtime

A browser is a program that knows how to render HTML documents

In the absence of style information, browser *applies basic default styling* (more on style later when we cover CSS)

JavaScript can interact with HTML to read and dynamically modify HTML code on the client side (within a browser)

# XHTML and HTML5

HTML was originally a small, clean tag set, suitable for simple text-document structures

- The development of commercial browsers led to specialized and often poorly-designed, browser-specific tags
- Browsers allowed sloppy, invalid syntax; documents became increasingly chaotic – not machine parse-able

XHTML: tightened up sloppy syntax, compatible with XML

HTML5: significant extensions to HTML capabilities

- still in process of being standardized
- more details to follow

# HyperText Markup Language

A "Web page" is a document that has been "marked up" using HTML "tags".

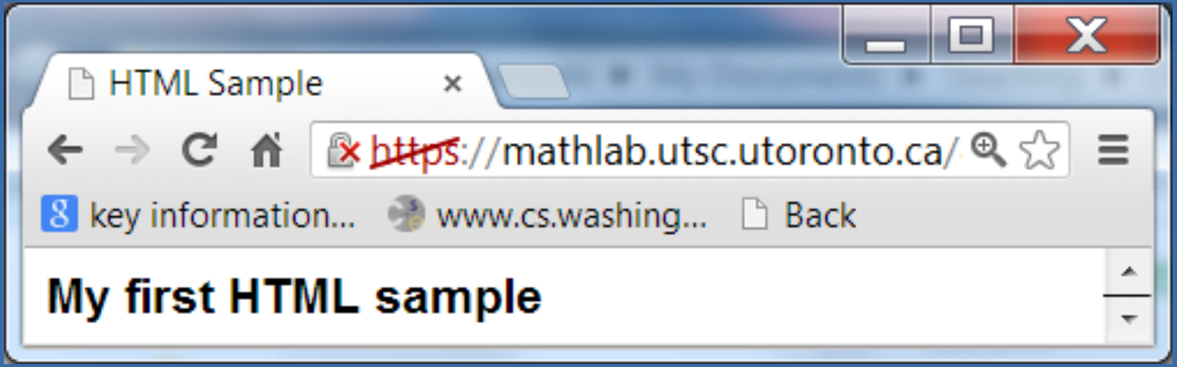
Tags are enclosed in angle brackets.

Must have matching open and close tag names.

Close tags have a / before the tag name.

Open and close tags enclose "elements".

```
<html>
  <head>
    <title>HTML Sample</title>
  </head>
  <body>
    <h3>My first HTML sample</h3>
  </body>
</html>
```



My first HTML sample

## HTML Elements

Angle-bracket-delimited tags mark the start and end of “elements”

```
<tag_name>element content</tag_name>
```

Elements can be nested inside of elements

```
<tag0>element0 content
   <tag1>nested element1 content
               <tag2>nested element2 content</tag2>
   more element1 content</tag1>
end of element0 content</tag0>
```

# Tag Structure and Content

Tags can have attributes that convey additional details about the tag;

```
<tagname attribute_name="attr_value" … >
```

```
<tag id="first" class="myclass">
```

Examples

- hypertext-link ("anchor") tags have an attribute that gives their URL:

  ```
  <a href="URL">link</a>
  ```

- image ("img") tags have an attribute that identifies the image URL, and text to display in case the image cannot be displayed:

  ```
  <img alt="text description" src="URL"/>
  ```

## Page Structure

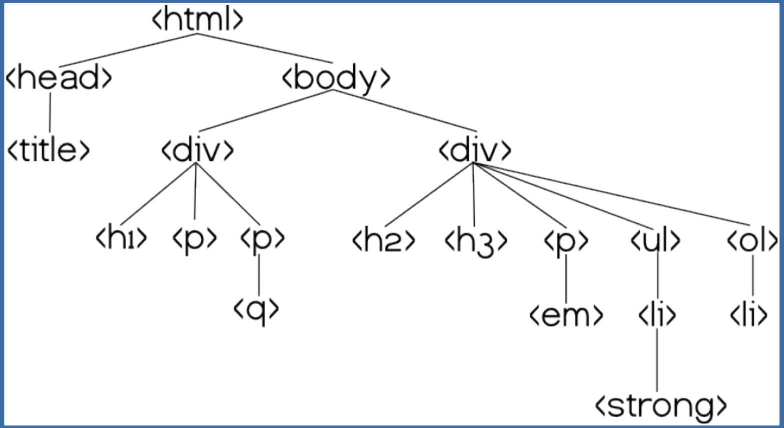
```
<html>
    <head>
        ...
    </head>
    <body>
      ...

    </body>
</html>
```

# Top-Level Structural Elements

- html – top-level (outermost) element – all other document elements are children of html
- head – overall document definitions, e.g. title which goes in browser title bar, style definitions (more with CSS later), client-side code definitions (more with JavaScript later)
- body

  – the content that makes up the "Web page"

  -- rendered by a browser, e.g. headings, paragraphs, lists



## `<title>`

- Required element within `<head>` element
- Provides a descriptive title that is displayed on a browser tab

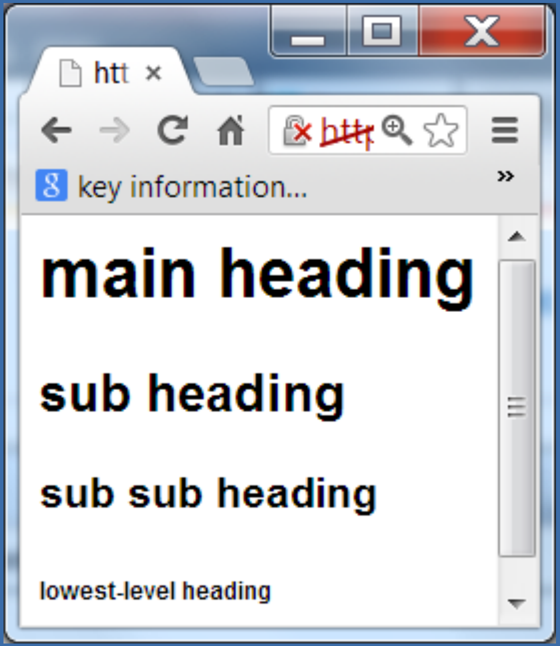
```
 <html>
    <head>
       <title>Tutorial 4 handout, Feb 10</title>
             …
    </head>
       …
</html>
```

# Headings (<h1> … <h6>)

Heading elements:

- Have line breaks above and below.
- *Lower numbers* correspond to higher-level (more important) headings

```
<h1>main heading</h1>
<h2>sub heading </h2>
<h3>sub sub heading</h3>
 …
<h6>lowest-level heading</h6>
```



main heading

sub heading

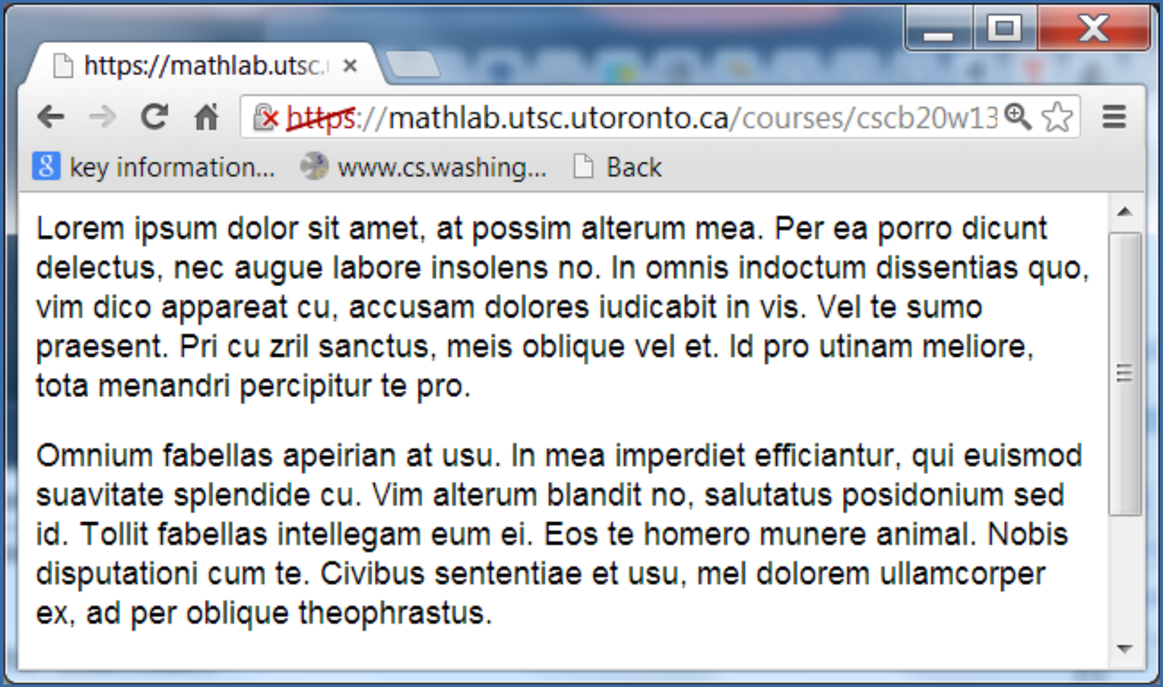
sub sub heading

lowest-level heading

# Paragraphs <p>

*Paragraphs* are block elements, set off from preceding and following elements by a line break.

```
<p>paragraph
text which may
contain many
sentences.</p>

<p>text of
next paragraph
</p>
```



Lorem ipsum dolor sit amet, at possim alterum mea. Per ea porro dicunt delectus, nec augue labore insolens no. In omnis indoctum dissentias quo, vim dico appareat cu, accusam dolores iudicabit in vis. Vel te sumo praesent. Pri cu zril sanctus, meis oblique vel et. Id pro utinam meliore, tota menandri percipitur te pro.

Omnium fabellas apeirian at usu. In mea imperdiet efficiantur, qui euismod suavitate splendide cu. Vim alterum blandit no, salutatus posidonium sed id. Tollit fabellas intellegam eum ei. Eos te homero munere animal. Nobis disputationi cum te. Civibus sententiae et usu, mel dolorem ullamcorper ex, ad per oblique theophrastus.

# Block and Inline Delimeters

Used in conjunction with class and id attributes.

Ties style sheet information and JavaScript behavior with document elements.

```
<div class="leftmenu”>
```

This is a generic “*block-level*” element (implicit line break).

```
<span id="someId">
```

Generic “*inline*” element (no line break)

No *visible rendering effect* is associated with either when used *without style information*.

# block and inline delimeters

```
<div class="section" id="forest-elephants">
  <h1>Forest elephants</h1>
  <p>In this section, we discuss the lesser
  known <span class="key-topic">forest
  elephants</span>.</p>
  ...this section continues...
  <div class="subsection" id="forest-
  habitat">
    <h2>Habitat</h2>
    <p>Forest elephants do not live in trees
    but among them.</p>
    ...this subsection continues...
  </div>
</div>
```

# hypertext links: the "anchor" element

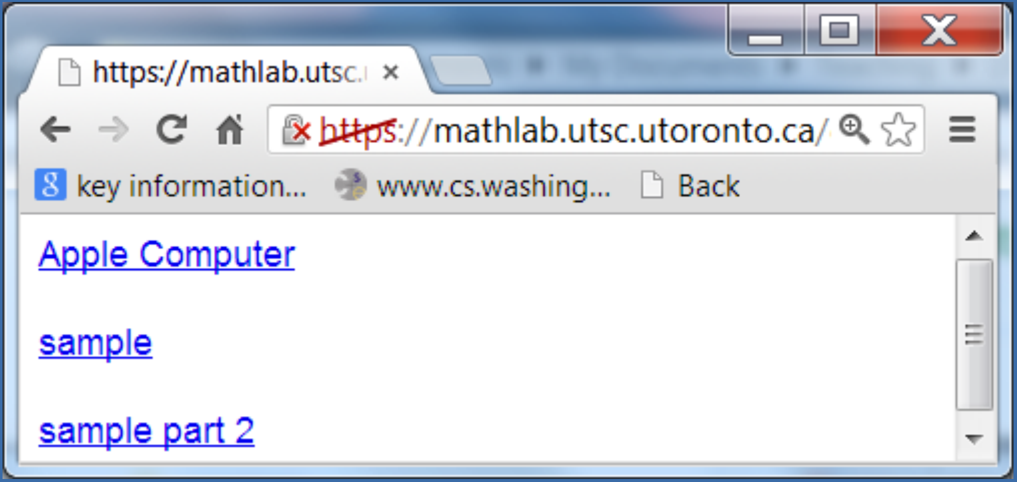
`<a>` tag with "`href`" attribute

Identifies the anchor-link URL

Can be absolute (http://...) or relative (path only)

And *offset* within a document, using "#" after path, and

```
<a href="http://www.apple.com">
Apple Computer</a>

<a href="sample.html">sample</a>

<a href="sample.html#part2">
sample part 2</a>
```



Apple Computer

sample

sample part 2

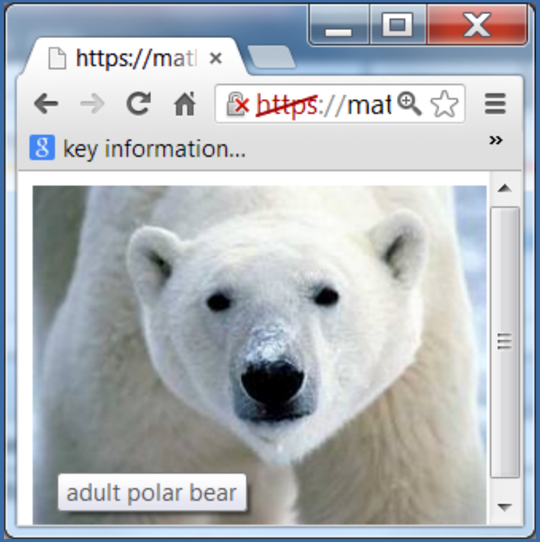
# Image Elements

*Not* part of original HTML definition

Key enabler for the Web's explosive growth in mid-late 1990's

Attributes:

- `src`="image_URL"
- `alt`="text description"
- `title`="mouseover description"



```
<img src="polar.jpg" alt="polar bear" title="adult
polar bear">
```

# Whitespace and Line Breaks

White space is generally ignored in the document layout

```
<p>This is the text of a very short
paragraph.</p>
```

would be rendered the same as:

```
<p>This is

the text           of a             very
short                 paragraph.</p>
```

Sometimes we want more control of layout:

`<br/>` forces a line break

```
<pre>text to be left unformatted</pre>
```

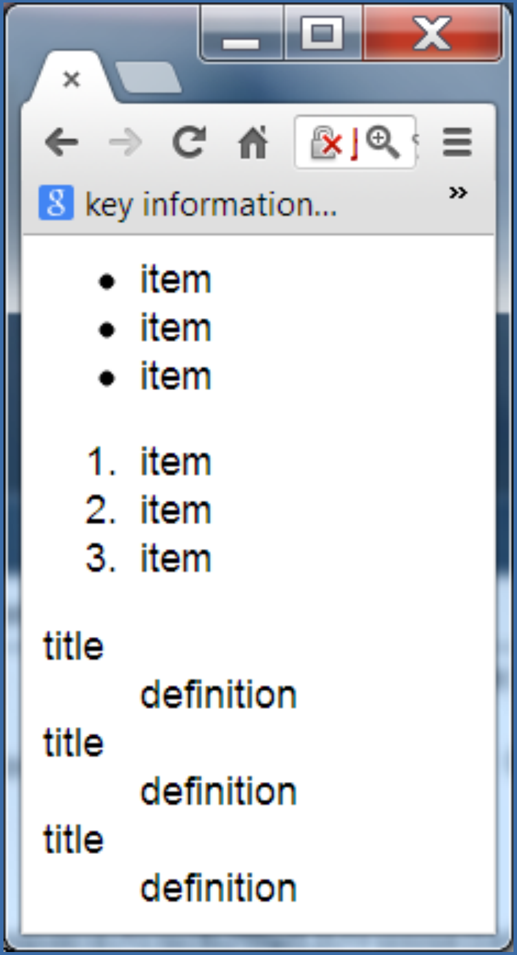
Beware that if the unformatted text includes tags, these will be interpreted as tags and not displayed.

# lists

3 basic types, with further variations via attributes

Unordered:

```
<ul><li>item</li>
   …
    <li>item</li>
</ul>
```



key information...

- item
- item
- item

1. item
2. item
3. item

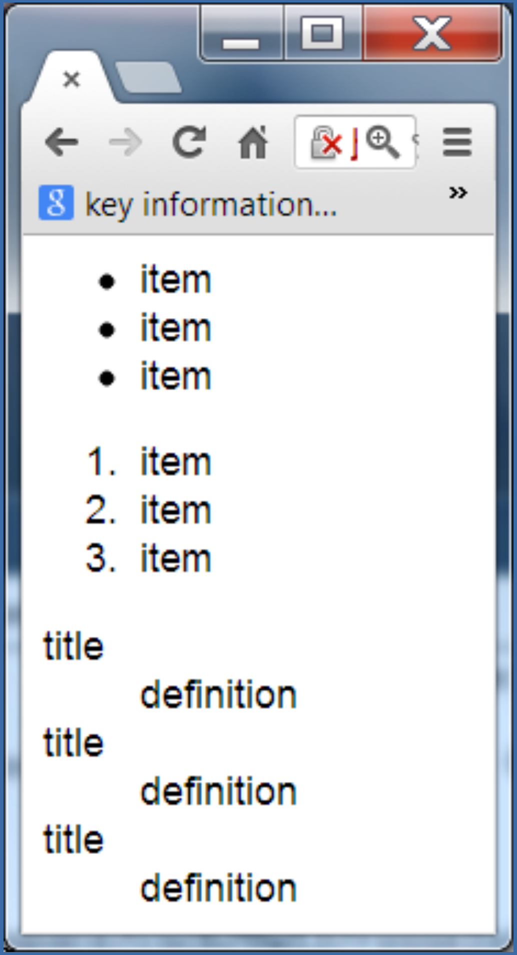
title
    definition
title
    definition
title
    definition

# lists

3 basic types, with further variations via attributes

Ordered:

```
<ol><li>item</li>
      …
      <li>item</li>
 </ol>
```


key information...

- item
- item
- item

1. item
2. item
3. item

title
    definition
title
    definition
title
    definition

# lists

3 basic types, with further variations via attributes

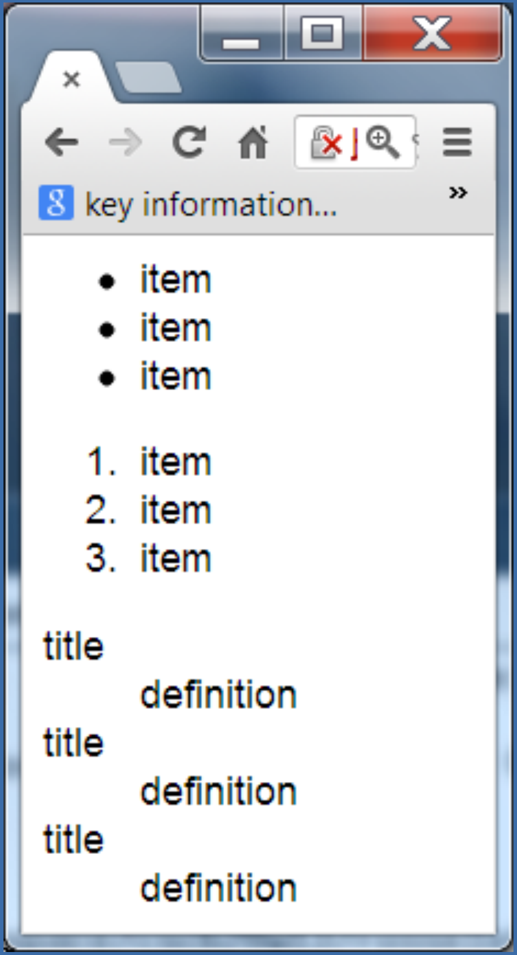
Definition (as in dictionary word-definition):

```
<dl>
     <dt>title</dt>
     <dd>definition</dd>
     …
</dl>
```

May be *arbitrarily nested*, e.g.

`<ul>` within `<ol>`,

`<ol>` within `<dl>`



- item
- item
- item

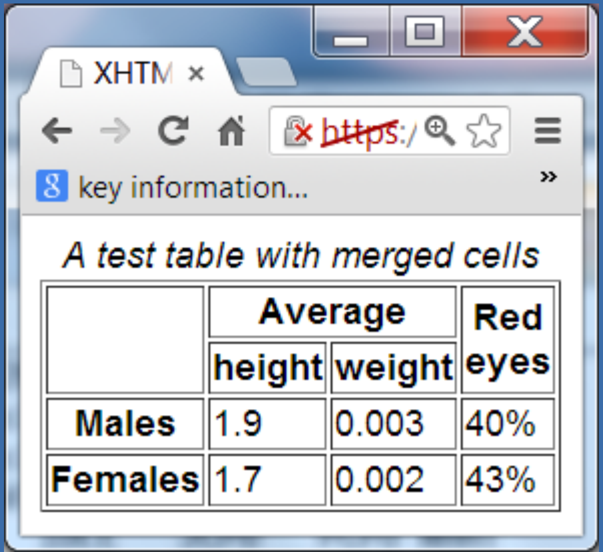
1. item
2. item
3. item

title
    definition
title
    definition
title
    definition

# tables

Widely used to control layout, but CSS is the recommended to handle layout.

In example, note use of spans (rowspan, colspan) to spread data across multiple rows/columns



A test table with merged cells

| | Average | | Red eyes |
|---|---|---|---|
| | height | weight | |
| Males | 1.9 | 0.003 | 40% |
| Females | 1.7 | 0.002 | 43% |

```
<table>
  <tr>
    <th rowspan="2"/>
    <th colspan="2">Average</th>
    <th rowspan="2">Red<br/>eyes</th>
  </tr>
  <tr>
    <th>height</th><th>weight</th>
  </tr>
  <tr>
    <th>Males</th>
    <td>1.9</td><td>0.003</td><td>40%</td>
  </tr>
  <tr>
    <th>Females</th>
    <td>1.7</td><td>0.002</td><td>43%</td>
  </tr>
</table>
```

•

# forms

Collect user input for:

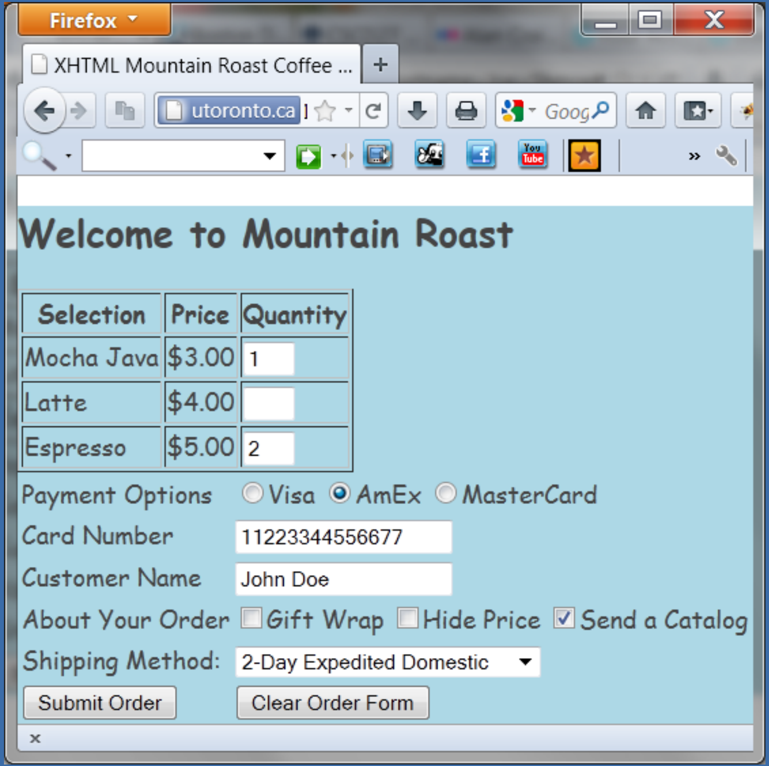
- processing by client-side scripts
- transmission to server-side programs

Built using widgets/controls:

- text boxes
- checkboxes
- pull-down select menus
- radio buttons
- submit and reset buttons

Each widget/control has a value

On submit, all widget values are collected together and sent to server



Welcome to Mountain Roast

| Selection | Price | Quantity |
|---|---|---|
| Mocha Java | $3.00 | 1 |
| Latte | $4.00 | |
| Espresso | $5.00 | 2 |

| | |
|---|---|
| Payment Options | Visa [✓ AmEx] MasterCard |
| Card Number | 11223344556677 |
| Customer Name | John Doe |
| About Your Order | Gift Wrap, Hide Price, ✓ Send a Catalog |
| Shipping Method: | 2-Day Expedited Domestic |
| Submit Order | Clear Order Form |

More about forms when we cover server-side programming

# <!-- comments -->

HTML has comments;  same syntax as XML

used to explain the non-obvious:

```
<!-- the following paragraph added to clarify
the meaning of the word "obfuscate" -->
```

introduce main sections of document

```
<!-- sidebar link display -->
```

enable parts of a page to be "hidden", e.g. to temporarily remove content, or for testing/debugging

```
<!-- <p>This paragraph covers a more-advanced
feature which will be explained later</p> -->
```

comments are *not nestable.*