HTML: Overview

Brief History of HTML

Did we mention that this will be brief?

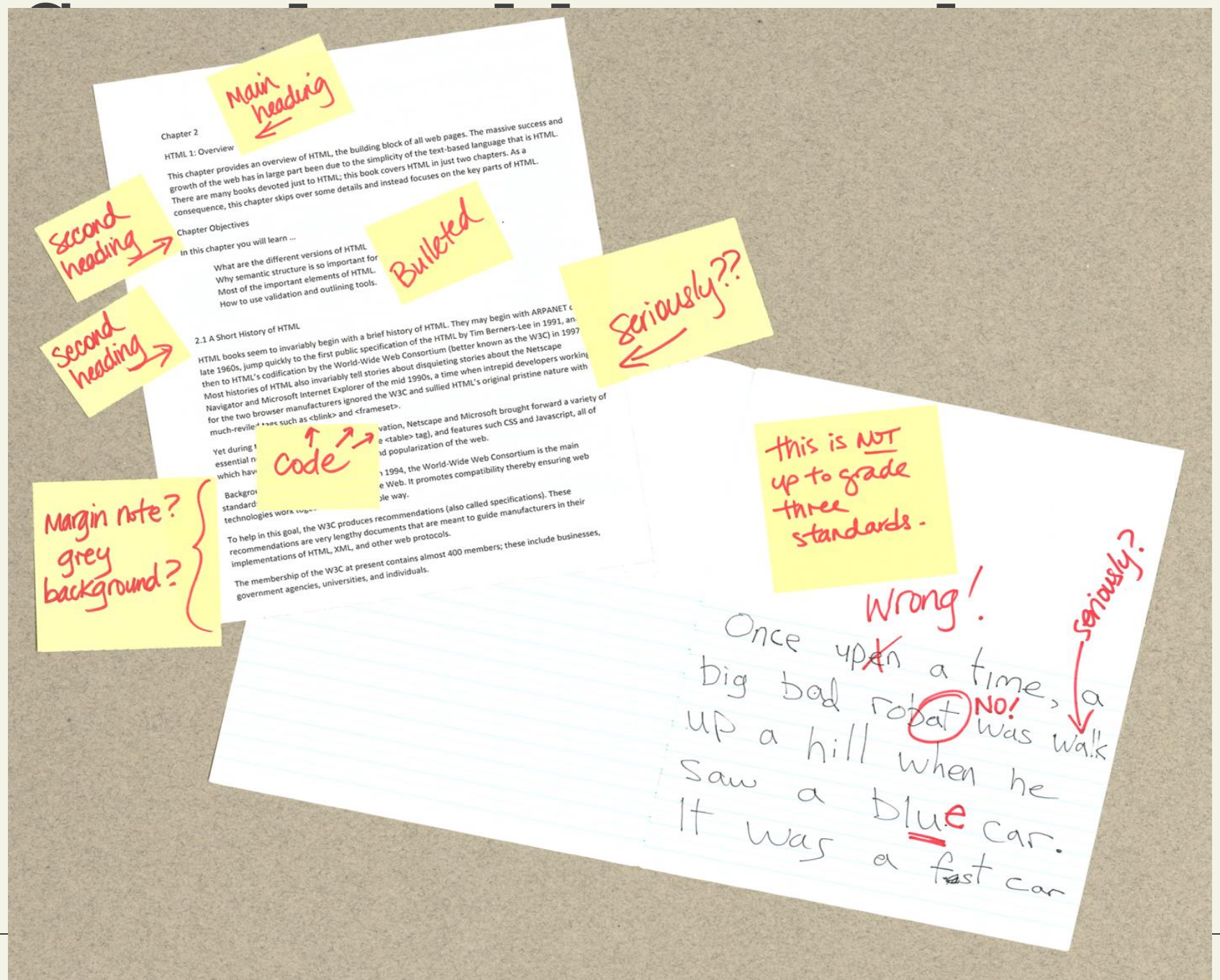
- First public specification of the HTML by Tim Berners-Lee in 1991
- HTML's codification by the World-Wide Web Consortium (better known as the **W3C**) in 1997.

HTML Syntax

What is a markup language?

HTML is defined as a **markup language**.

- A markup language is simply a way of annotating a document in such a way to make the annotations distinct from the text being annotated.
- At its simplest, **markup** is a way to indicate *information about the content*
- This “information about content” in HTML is implemented via textual **tags** (aka elements).


Main heading
Chapter 2
HTML 1: Overview
This chapter provides an overview of HTML, the building block of all web pages. The massive success and growth of the web has in large part been due to the simplicity of the text-based language that is HTML. There are many books devoted just to HTML; this book covers HTML in just two chapters. As a consequence, this chapter skips over some details and instead focuses on the key parts of HTML.
second heading
Chapter Objectives
In this chapter you will learn ...
What are the different versions of HTML
Why semantic structure is so important for
Most of the important elements of HTML.
How to use validation and outlining tools.
Bulleted
second heading
2.1 A Short History of HTML
HTML books seem to invariably begin with a brief history of HTML. They may begin with ARPANET
late 1960s, jump quickly to the first public specification of the HTML by Tim Berners-Lee in 1991, an
then to HTML's codification by the World-Wide Web Consortium (better known as the W3C) in 1997
Most histories of HTML also invariably tell stories about disquieting stories about the Netscape
Navigator and Microsoft Internet Explorer of the mid 1990s, a time when intrepid developers working
for the two browser manufacturers ignored the W3C and sullied HTML's original pristine nature with
much-revile
such as <blink> and <frameset>.
Yet during
essential
which hav
vation, Netscape and Microsoft brought forward a variety of
e <table> tag), and features such CSS and Javascript, all of
d popularization of the web.
Code
Backgro
standard
technologies work
n 1994, the World-Wide Web Consortium is the main
e Web. It promotes compatibility thereby ensuring web
le way.
To help in this goal, the W3C produces recommendations (also called specifications). These recommendations are very lengthy documents that are meant to guide manufacturers in their implementations of HTML, XML, and other web protocols.
The membership of the W3C at present contains almost 400 members; these include businesses, government agencies, universities, and individuals.
Margin note? grey background?
Seriously??
this is NOT up to grade three standards.
Wrong!
Once upon a time, a
big bad robot was walk
up a hill when he
saw a blue car.
It was a fast car
NO!
Seriously?

Markup

What is it again?

At its simplest, **markup** is a way to indicate *information about the content*

- This “information about content” in HTML is implemented via **tags**.
- The markup in the previous slide consists of the red text and the various circles and arrows on the one page, and the little yellow sticky notes on the other.
- HTML does the same thing but uses textual tags.

What is the W3C?

Standards

The W3C is the main standards organization for the World Wide Web.

To promotes compatibility the W3C produces **recommendations** (also called **specifications**).

In 1998, the W3C turned its attention to a new specification called XHTML 1.0, which was a version of HTML that used stricter **XML** (Extensible Markup Language) syntax rules.

XML Overview

XML is a markup language, but unlike HTML, XML can be used to mark up any type of data.

One of the key benefits of XML data is that as plain text, it can be read and transferred between applications and different operating systems as well as being human-readable and understandable as well.

XML is not only used on the web server and to communicate asynchronously with the browser, but is also used as a data interchange format for moving information between systems

Well Formed XML

For a document to be **well-formed XML**, it must follow the syntax rules for XML:

- Element names are composed of any of the valid characters (most punctuation symbols and spaces are not allowed) in XML.
- Element names can’t start with a number.
- There must be a single-root element. A **root element** is one that contains all the other elements; for instance, in an HTML document, the root element is <html>.
- All elements must have a closing element (or be self-closing).
- Elements must be properly nested.
- Elements can contain attributes.
- Attribute values must always be within quotes.
- Element and attribute names are case sensitive.

Well Formed XML

Sample Document

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<art>
  <painting id="290">
    <title>Balcony</title>
    <artist>
      <name>Manet</name>
      <nationality>France</nationality>
    </artist>
    <year>1868</year>
    <medium>Oil on canvas</medium>
  </painting>
  <painting id="192">
    <title>The Kiss</title>
    <artist>
      <name>Klimt</name>
      <nationality>Austria</nationality>
    </artist>
    <year>1907</year>
    <medium>Oil and gold on canvas</medium>
  </painting>
  <painting id="139">
    <title>The Oath of the Horatii</title>
    <artist>
      <name>David</name>
      <nationality>France</nationality>
    </artist>
    <year>1784</year>
    <medium>Oil on canvas</medium>
  </painting>
</art>
```

LISTING 17.1 Sample XML document

Valid XML

Requires a DTD

A **valid XML** document is one that is well formed and whose element and content conform to the rules of either its document type definition (DTD) or its schema.

A DTD tells the XML parser which elements and attributes to expect in the document as well as the order and nesting of those elements.

A DTD can be defined within an XML document or within an external file.

Data Type Definition

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE art [
<!ELEMENT art (painting*)>
<!ELEMENT painting (title,artist,year,medium)>
<!ATTLIST painting id CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT artist (name,nationality)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT nationality (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT medium (#PCDATA)>
]>
<art>
...
</art>
```

LISTING 17.2 Example DTD

HTML5

Three main aims

HTML5 (October 2014) has three main aims:

- Specify unambiguously how browsers should deal with invalid markup.
- Provide an open, non-proprietary programming framework (via Javascript) for creating rich web applications.
- Be backwards compatible with the existing web.

HTML SYNTAX

Elements and Attributes

More syntax

HTML documents are composed of textual content and HTML elements.

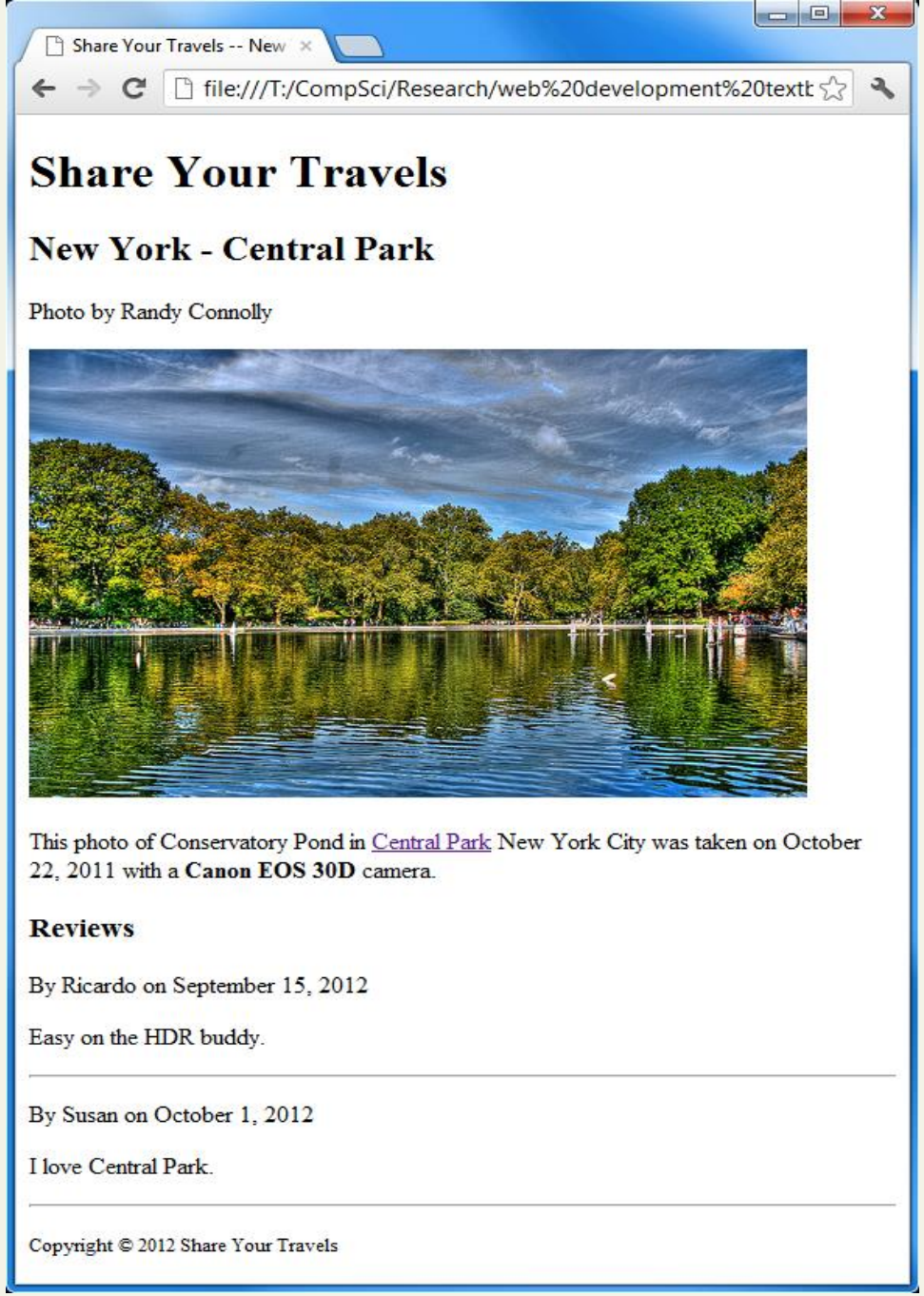
An **HTML element** can contain text, other elements, or be empty. It is identified in the HTML document by tags.

HTML elements can also contain attributes. An **HTML attribute** is a name=value pair that provides more information about the HTML element.

What HTML lets you do

- Insert images using the **<img>** tag
- Create links with the **<a>** tag
- Create lists with the **<ul>, ** and **<li>** tags
- Create headings with **<h1>, <h2>, …, <h6>**
- Define metatdata with **<meta>** tag
- And much more…

Sample Document



```
<body>
   <h1>Share Your Travels</h1>
   <h2>New York - Central Park</h2>
   <p>Photo by Randy Connolly</p>
   <p>This photo of Conservatory Pond in
      <a href="http://www.centralpark.com/">Central Park</a>
      New York City was taken on October 22, 2011 with a
      <strong>Canon EOS 30D</strong> camera.
   </p>
   <img src="images/central-park.jpg" alt="Central Park" />

   <h3>Reviews</h3>
   <div>
      <p>By Ricardo on <time>September 15, 2012</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>

   <div>
      <p>By Susan on <time>October 1, 2012</time></p>
      <p>I love Central Park.</p>
   </div>

   <p><small>Copyright &copy; 2012 Share Your Travels</small></p>
</body>
```

Elements and Attributes

Opening Tag

Closing Tag

```
<a href="http://www.centralpark.com">Central Park</a>
```

Element Name

Attribute

Content

May be text or other HTML elements

Trailing Slash

Example empty element

```
<br />
```

Element Name

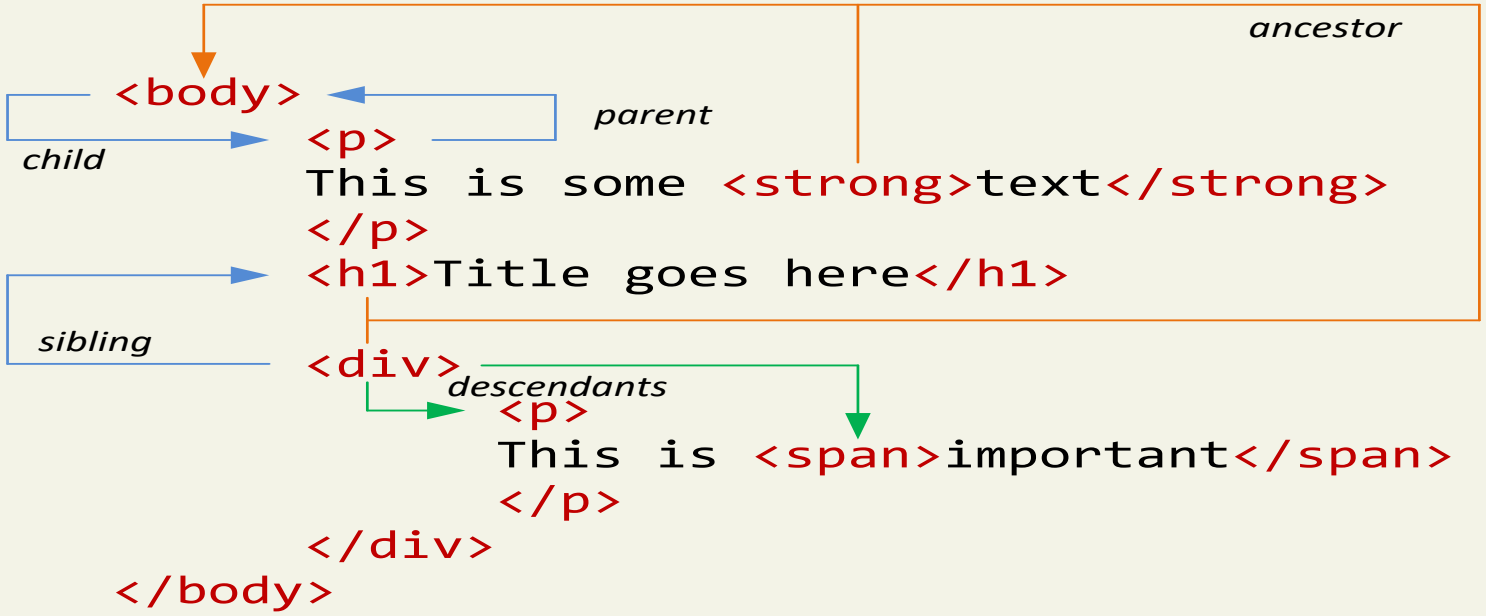
Nesting HTML elements

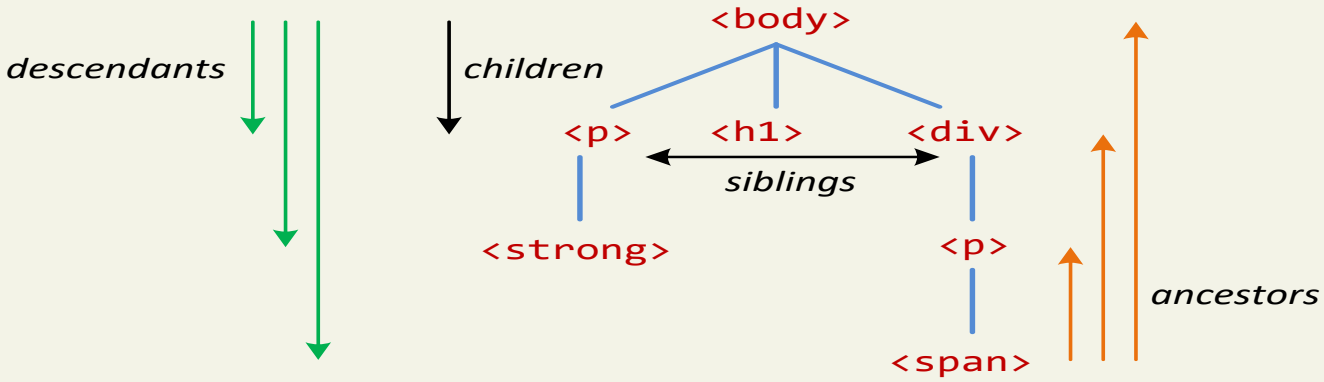
Often an HTML element will contain other HTML elements.

In such a case, the container element is said to be a parent of the contained, or child, element.

Any elements contained within the child are said to be **descendents** of the parent element; likewise, any given child element, may have a variety of **ancestors**.

Hierarchy of elements





Nesting HTML elements

In order to properly construct a hierarchy of elements, your browser expects each HTML nested element to be properly nested.

That is, a child's ending tag must occur before its parent's ending tag.

Correct Nesting

```
<h1>Share Your <strong>Travels</strong></h1>
```

```
<h1>Share Your <strong>Travels</h1></strong>
```

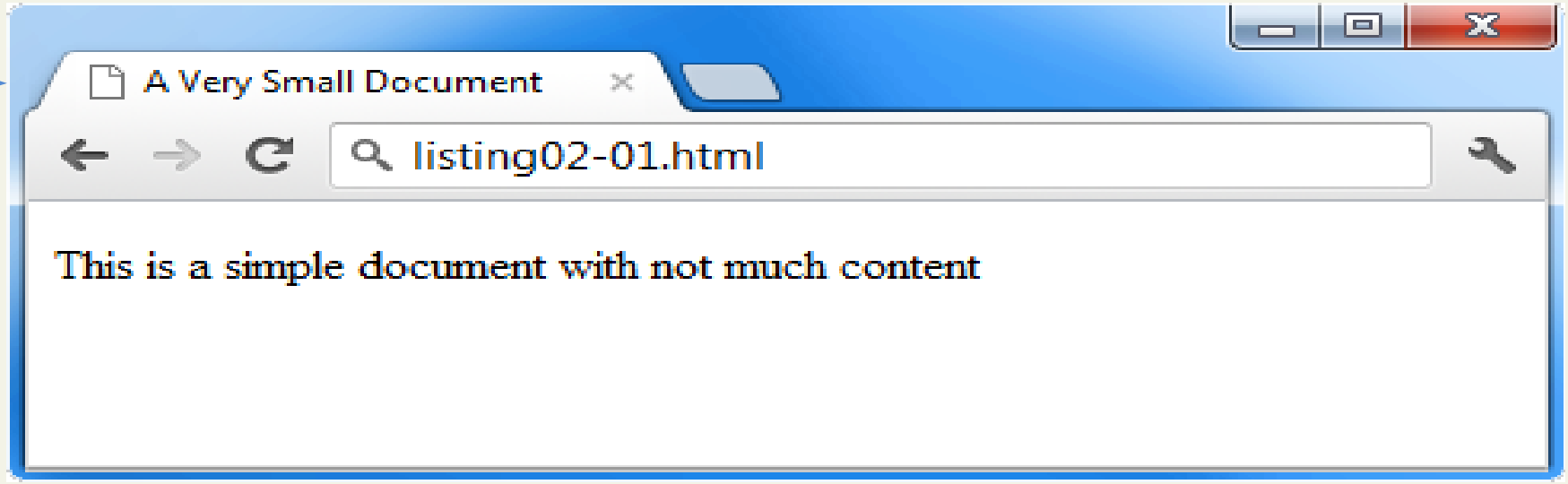
Incorrect Nesting

STRUCTURE OF HTML

Simplest HTML document

1

```
<!DOCTYPE html>
<title>A Very Small Document</title>
<p>This is a simple document with not much content</p>
```



The <title> element (Item 1) is used to provide a broad description of the content. The title is not displayed within the browser window. Instead, the title is typically displayed by the browser in its window and/or tab.

A more complete document

```
1  <!DOCTYPE html>
2  <html>
3    <head lang="en">
       <meta charset="utf-8">                                  5
       <title>Share Your Travels -- New York - Central Park</title>
       <link rel="stylesheet" href="css/main.css">             6
       <script src="js/html5shiv.js"></script>                 7
     </head>
4    <body>
       <h1>Main heading goes here</h1>
       ...
     </body>
   </html>
```

DOCTYPE

(short for **Document Type Definition**)

1. Tells the browser (or any other client software that is reading this HTML document) what type of document it is about to process.

Notice that it does not indicate what version of HTML is contained within the document: it only specifies that it contains HTML.

```
1  <!DOCTYPE html>
2  <html>
3     <head lang="en">
5        <meta charset="utf-8">
         <title>Share Your Travels -- New York - Central Park</title>
6        <link rel="stylesheet" href="css/main.css">
7        <script src="js/html5shiv.js"></script>
      </head>
4     <body>
         <h1>Main heading goes here</h1>
         ...
      </body>
      </html>
```

Head and Body

2. The <html> element is sometimes called the **root element** as it contains all the other HTML elements in the document.

   HTML pages are divided into two sections: the **head** and the **body**, which correspond to the <head> and <body> elements.

3. The head contains descriptive elements *about* the document

4. The body contains content that will be displayed by the browser.

```
1  <!DOCTYPE html>
2  <html>
3     <head lang="en">
5        <meta charset="utf-8">
         <title>Share Your Travels -- New York - Central Park</title>
6        <link rel="stylesheet" href="css/main.css">
7        <script src="js/html5shiv.js"></script>
      </head>
4     <body>
         <h1>Main heading goes here</h1>
         ...
      </body>
   </html>
```

Inside the head

There are no brains

You will notice that the <head> element contains a variety of additional elements.

5 The first of these is the <meta> element. Our example declares that the character encoding for the document is UTF-8.

```
1  <!DOCTYPE html>
2  <html>
3     <head lang="en">
         <meta charset="utf-8">                                  5
         <title>Share Your Travels -- New York - Central Park</title>
         <link rel="stylesheet" href="css/main.css">             6
         <script src="js/html5shiv.js"></script>                 7
     </head>
4    <body>
         <h1>Main heading goes here</h1>
         ...
     </body>
     </html>
```

Inside the head

No brains but metas, styles and javascripts

6. Our example specifies an external CSS style sheet file that is used with this document.

7. It also references an external Javascript file.

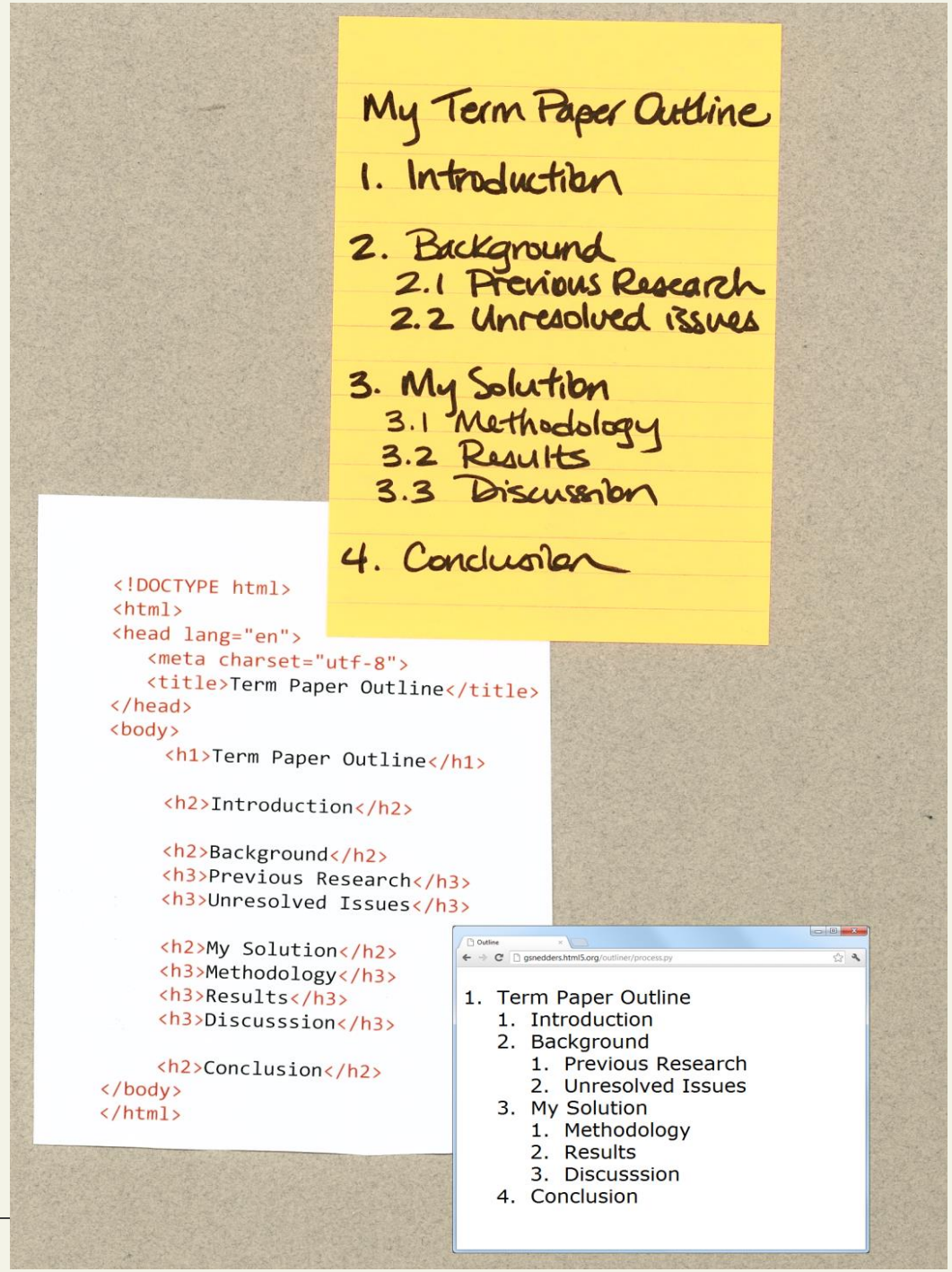
```
1  <!DOCTYPE html>
2  <html>
3     <head lang="en">
         <meta charset="utf-8">                                  5
         <title>Share Your Travels -- New York - Central Park</title>
         <link rel="stylesheet" href="css/main.css">             6
         <script src="js/html5shiv.js"></script>                 7
      </head>
4     <body>
         <h1>Main heading goes here</h1>
         ...
      </body>
   </html>
```

Headings

<h1>, <h2>, <h3>, etc

HTML provides six levels of heading (**h1, h2, h3**, …), with the higher heading number indicating a heading of less importance.

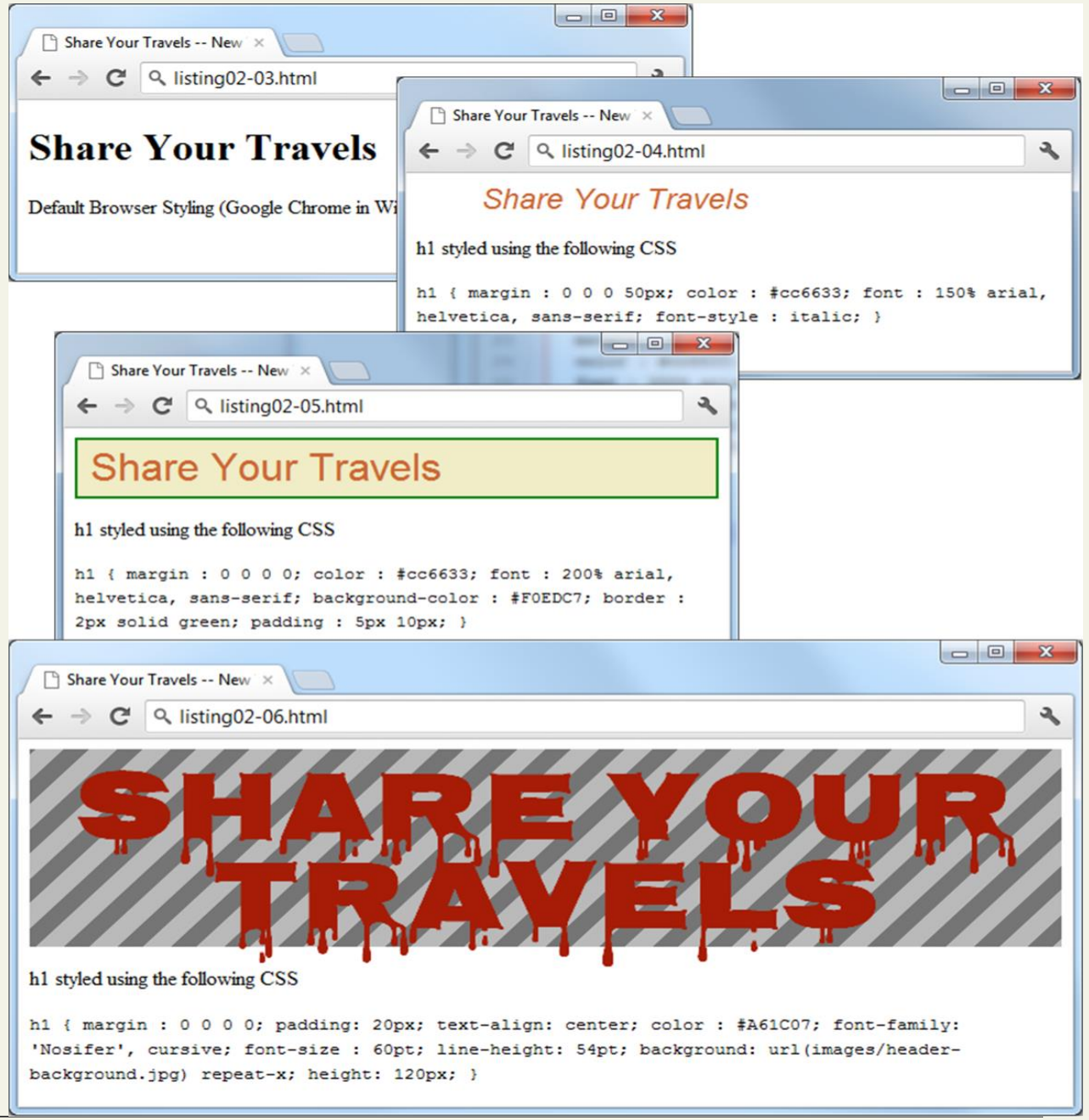
Headings are an essential way for document authors use to show their readers the structure of the document.



Headings

The browser has its own default styling for each heading level.

However, these are easily modified and customized via CSS.



Headings

Be semantically accurate

In practice, specify a heading level that is semantically accurate.

Do not choose a heading level because of its default presentation

- e.g., choosing <h3> because you want your text to be bold and 16pt

Rather, choose the heading level because it is appropriate

- e.g., choosing <h3> because it is a third level heading and not a primary or secondary heading

Paragraphs
<p>

Paragraphs are the most basic unit of text in an HTML document.

Notice that the **<p>** tag is a container and can contain HTML and other **inline HTML elements**

- **inline HTML elements** refers to HTML elements that do not cause a paragraph break but are part of the regular “flow” of the text.
- **Block HTML elements** build a block around themselves, causing a break in the flow oh the text (e.g., paragraph, table)
- Composition: Block elements can contain block and inline elements, while inline elements can contain only inline elements

Divisions

This ** ** tag is also a container element and is used to create a logical grouping of content

- The element has no intrinsic presentation.
- It is frequently used in contemporary CSS-based layouts to mark out sections.

Using div elements

HTML5 has a variety of new semantic elements (which we will examine later) that can be used to reduce somewhat the confusing mass of div within divs within divs that is so typical of contemporary web design.

```
<!DOCTYPE html>
▼<html lang="en-US">
  <!-- Developed by Digital Cavalry 2012 (http://themeforest.net/user/DigitalCavalry) -->
  ▶<head>…</head>
  ▼<body class="home page page-id-36 page-template page-template-content-builder-php">
    ▼<div class="dc-body-wrapper">
      ▼<div class="dc-body-inner-wrapper">
          <a id="dc-site-top-anchor" name="dc-site-top-anchor"></a>
        ▶<div class="dc-site-header">…</div>
        ▶<div id="dc-primary-theme-menu-wrapper-wrapper">…</div>
        ▶<select id="dc-primary-theme-menu-responsive">…</select>
        ▼<div class="dc-primary-wrapper">
          ▼<div class="dc-secondary-wrapper">
              <div class="dc-wp-breadcrumb-navigation-empty"></div>
            ▼<div class="dc-page-seo-wrapper dc-layout-full-width">
              ▼<div class="dc-page-content">
                ▼<div class="dc-content-builder-wrapper">
                  ▼<div class=" dc-sixteen dc-columns  " style="padding-top:0px;padding-bottom:
                  20px;float:left;">
                    ▼<div class="dc-over-wrapper" style="padding-right:0px;padding-left:0px;">
                      ▼<div class="dc-basic-slider" style="margin-bottom:0px;">
                        ▶<div class="slider-options">…</div>
                        ▼<div class="inner-wrapper">
                          ▶<ul>…</ul>
                            <div class="nav-next-btn" style="display: none;"></div>
                            <div class="nav-prev-btn" style="display: none;"></div>
                          </div>
                        ▼<div class="nav-pager">
                            <div class="page"></div>
                            <div class="page"></div>
                            <div class="page page-on"></div>
                            <div class="page"></div>
                            <div class="page"></div>
```

Links

<a>

Links are created using the **<a>** element (the “a” stands for anchor).

A link has two main parts: the destination and the label.

```
<a href="http://www.centralpark.com">Central Park</a>
```

Destination

Label (text)

```
<a href="index.html"><img src="logo.gif" /></a>
```

Label (image)

Different link destinations

You can use the anchor element to create a wide range of links:

Link to external site

```
<a href="http://www.centralpark.com">Central Park</a>
```

Link to resource on external site

```
<a href="http://www.centralpark.com/logo.gif">Central Park</a>
```

Link to another page on same site as this page

```
<a href="index.html">Home</a>
```

Link to another place on the same page

```
<a href="#top">Go to Top of Document</a>
```

Different link destinations

Link to specific place on another page

```
<a href="productX.html#reviews">Reviews for product X</a>
```

Link to email

```
<a href="mailto://person@somewhere.com">Someone</a>
```

Link to javascript function

```
<a href="javascript://OpenAnnoyingPopup();">See This</a>
```

Link to telephone (automatically dials the number when user clicks on it using a smartphone browser)

```
<a href="tel:+18009220579">Call toll free (800) 922-0579</a>
```

URL Absolute Referencing

For external resources

When referencing a page or resource on an external site, a full **absolute reference** is required: that is,

- the protocol (typically, http://),
- the domain name,
- any paths, and then finally
- the file name of the desired resource.

URL Relative Referencing

An essential skill

We also need to be able to successfully reference files within our site.

This requires learning the syntax for so-called **relative referencing**.

If the URL does not include the “http://” then the browser will request the current server for the file.

For these situations, a relative pathname (following UNIXi conventions) is required along with the filename.

Inline Text Elements

Do not disrupt the flow

Inline elements do not disrupt the flow of text (i.e., cause a line break).

HTML5 defines over 30 of these elements.

e.g., <a>, <br>, ,

Images

While the **<img>** tag is the oldest method for displaying an image, it is not the only way.

For purely decorative images, such as background gradients and patterns, logos, border art, and so on, it makes semantic sense to keep such images out of the markup and in CSS where they more rightly belong.

But when the images are content, such as in the images in a gallery or the image of a product in a product details page, then the <img> tag is the semantically appropriate approach.

Images

Specifies the URL of the image to display (note: uses standard relative referencing)

Text in title attribute will be displayed in a popup tool tip when user moves mouse over image.

```
<img src="images/central-park.jpg" alt="Central Park"  title="Central Park" width="80" height="40" />
```

Text in alt attribute provides a brief description of image's content for users who are unable to see it.

Specifies the width and height of image in pixels.

Lists

HTML provides three types of lists

**Unordered lists **. Collections of items in no particular order; these are by default rendered by the browser as a bulleted list.

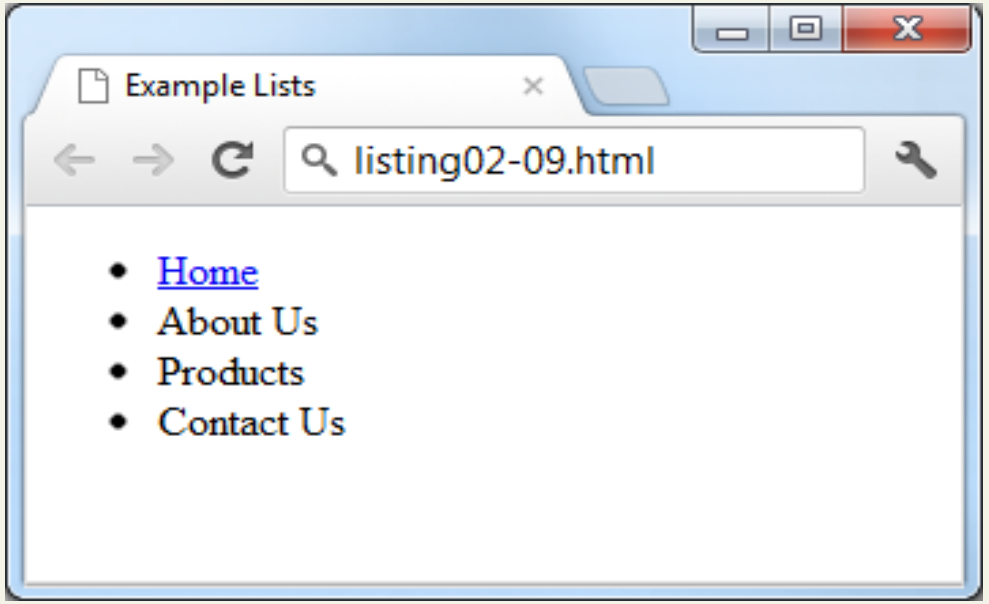
**Ordered lists **. Collections of items that have a set order; these are by default rendered by the browser as a numbered list.

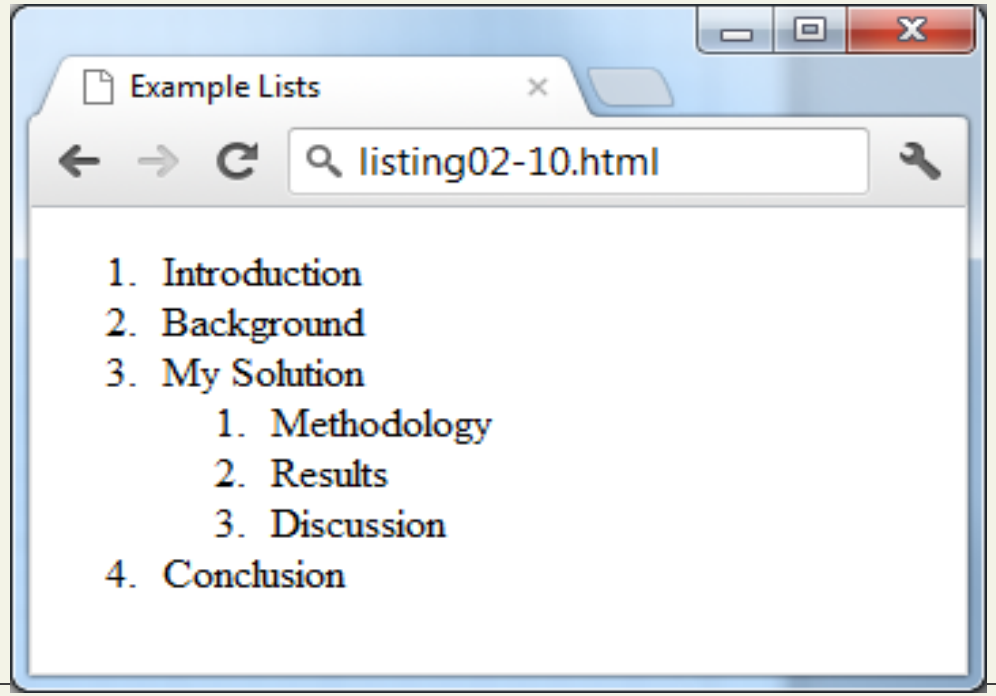
Definition lists <dt>. Collection of name <dt> and definition <dd> pairs. These tend to be used infrequently. Perhaps the most common example would be a FAQ list.

Notice that the list item element can contain other HTML elements

```
<ul>
   <li><a href="index.html">Home</a></li>
   <li>About Us</li>
   <li>Products</li>
   <li>Contact Us</li>
</ul>
```

```
<ol>
   <li>Introduction</li>
   <li>Background</li>
   <li>My Solution</li>
   <li>
      <ol>
         <li>Methodology</li>
         <li>Results</li>
         <li>Discussion</li>
      </ol>
   </li>
   <li>Conclusion</li>
</ol>
```





HTML Tables

A grid of cells

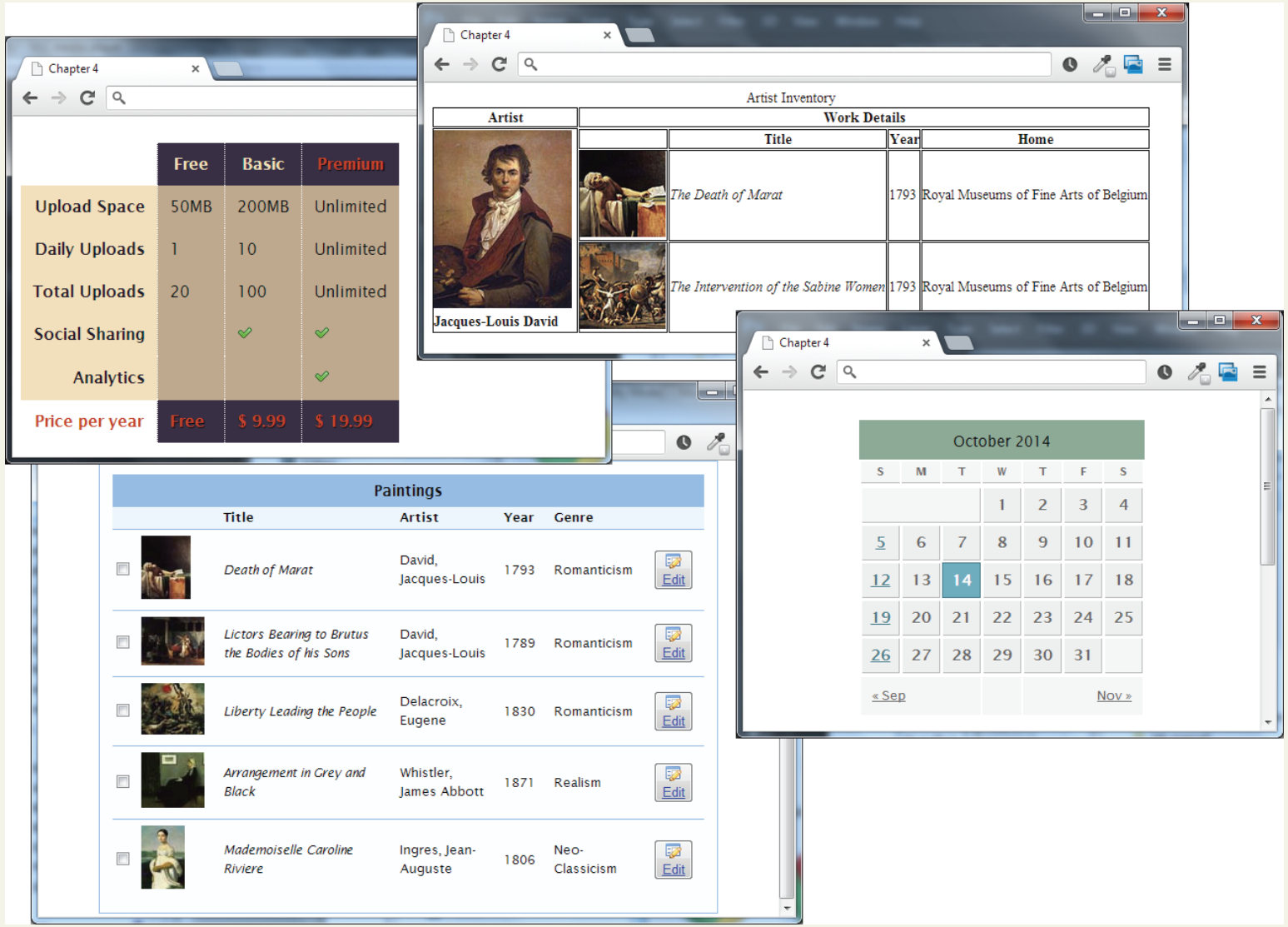
A table in HTML is created using the **<table>** element

Tables can be used to display:

- Many types of content
  - Calendars, financial data, lists, etc…
- Any type of data
  - Images
  - Text
  - Links
  - Other tables

HTML Tables

Example usages



Tables Basics

Rows and cells

- an HTML **<table>** contains any number of rows (**<tr>**)
- each row contains any number of table data cells (**<td>**)
- Content goes inside of **<td></td>** tags

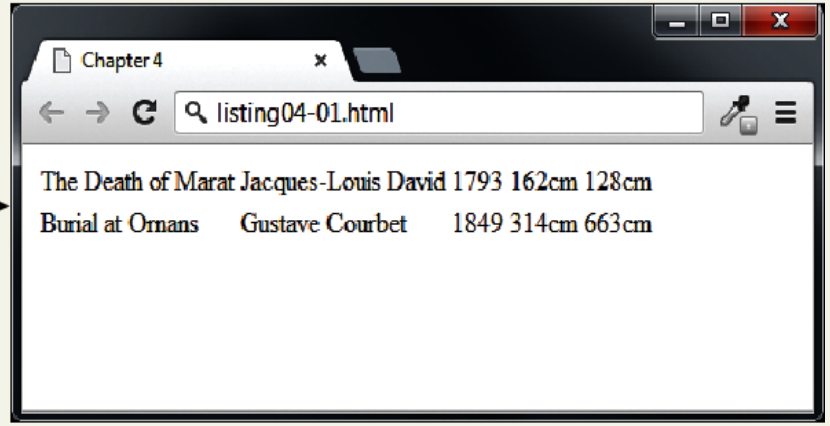
```
<table>
        <tr>
                <td>The Death of Marat</td>
        </tr>
</table>
```

content

<table>

<tr>	The Death of Marat <td>	Jacques-Louis David <td>	1793 <td>	162cm <td>	128cm <td>
<tr>	Burial at Ornans <td>	Gustave Courbet <td>	1849 <td>	314cm <td>	663cm <td>

```
<table>
    <tr>
        <td>The Death of Marat</td>
        <td>Jacques-Louis David</td>
        <td>1793</td>
        <td>162cm</td>
        <td>128cm</td>
    </tr>
    <tr>
        <td>Burial at Ornans</td>
        <td>Gustave Courbet</td>
        <td>1849</td>
        <td>314cm</td>
        <td>663cm</td>
    </tr>
 </table>
```

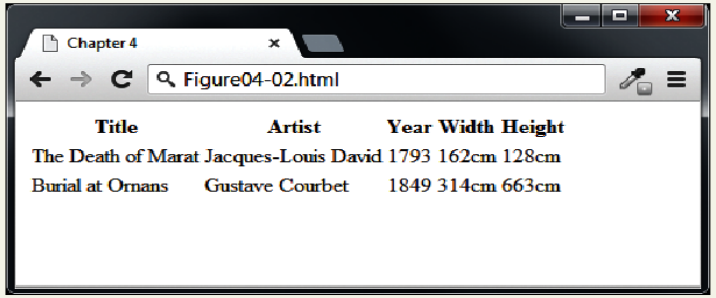


<table>

<tr>	Title <th>	Artist <th>	Year <th>	Width <th>	Height <th>
<tr>	The Death of Marat <td>	Jacques-Louis David <td>	1793 <td>	162cm <td>	128cm <td>
<tr>	Burial at Ornans <td>	Gustave Courbet <td>	1849 <td>	314cm <td>	663cm <td>

```
<table>
   <tr>
      <th>Title</th>
      <th>Artist</th>
      <th>Year</th>
      <th>Width</th>
      <th>Height</th>
   </tr>
   <tr>
      <td>The Death of Marat</td>
      <td>Jacques-Louis David</td>
      <td>1793</td>
      <td>162cm</td>
      <td>128cm</td>
   </tr>
   <tr>
      <td>Burial at Ornans</td>
      <td>Gustave Courbet</td>
      <td>1849</td>
      <td>314cm</td>
      <td>663cm</td>
   </tr>
</table>
```

th



Spanning Rows and Columns

<table>

<tr>	Title <th>	Artist <th>	Year <th>	Size (width x height) <th colspan=2>	
<tr>	The Death of Marat <td>	Jacques-Louis David <td>	1793 <td>	162cm <td>	128cm <td>
<tr>	Burial at Ornans <td>	Gustave Courbet <td>	1849 <td>	314cm <td>	663cm <td>

use the **colspan or rowspan attributes**

Notice that this row now only has four cell elements.

```
<table>
   <tr>
      <th>Title</th>
      <th>Artist</th>
      <th>Year</th>
      <th colspan="2">Size (width x height)</th>
   </tr>
   <tr>
      <td>The Death of Marat</td>
      <td>Jacques-Louis David</td>
      <td>1793</td>
      <td>162cm</td>
      <td>128cm</td>
   </tr>
   ...
</table>
```

Character Entities

These are special characters for symbols for which there is either no way easy way to type in via a keyboard (such as the copyright symbol or accented characters) or which have a reserved meaning in HTML (for instance the “<” or “>” symbols).

They can be used in an HTML document by using the entity name or the entity number.

e.g.,   and ©

SEMANTIC MARKUP

Semantic Markup

What does it mean?

Over the past decade, a strong and broad consensus has grown around the belief that HTML documents should **only** focus on the structure of the document.

Information about how the content should look when it is displayed in the browser is best left to CSS (Cascading Style Sheets).

Semantic Markup

As a consequence, beginning HTML authors are often advised to create **semantic HTML** documents.

That is, an HTML document should not describe how to visually present content, but only describe its content's structural semantics or meaning.

Structure

Structure is a vital way of communicating information in paper and electronic documents.

All of the tags that we will examine in this presentation are used to describe the basic structural information in a document, such as articles, headings, lists, paragraphs, links, images, navigation, footers, and so on.

Semantic Markup

Its advantages

Eliminating presentation-oriented markup and writing semantic HTML markup has a variety of important advantages:

Maintainability. Semantic markup is easier to update and change than web pages that contain a great deal of presentation markup.

Faster. Semantic web pages are typically quicker to author and faster to download.

Accessibility. Visiting a web page using voice reading software can be a very frustrating experience if the site does not use semantic markup.

Search engine optimization. Semantic markup provides better instructions for search engines: it tells them what things are important content on the site.

HTML5 Semantic Elements

Why are they needed?

One substantial problem with modern, pre-HTML5 semantic markup:

- most complex web sites are absolutely packed solid with elements.

Unfortunately, all these elements can make the resulting markup confusing and hard to modify.

Developers typically try to bring some sense and order to the chaos by using id or class names that provide some clue as to their meaning.

Problem

```
<body>
  <div id="header">
    <div id="logo-headings">
      ...
    </div>
   ...
    <div id="top-navigation">
      ...
    </div>
  </div>
  <div id="main">
     <div id="left-navigation">
       ...
     </div>
     <div class="content">
       <div class="story">
         ...
       </div>
       <div class="story">
         ...
         <div class="story-photo">
           <img ... class="blog-photo"/>
           <p classs="photo-caption">...
         </div>
       </div>
       <div class="related-stuff-on-right">
         ...
       </div>
     </div>
     <div class="content">
       ...
     </div>
  </div>
  <div id="footer">
    ...
  </div>
</body>
```

1 <header>

2 <hgroup>

3 <nav>

4

5 <section>

6 <article>

7 <figure>

8 <figcaption>

9 <aside>

10 <footer>

Solution

```
<body>
  <header>
    <hgroup>
      ...
    </hgroup>
    ...
    <nav>
      ...
    </nav>
  </header>
  <div id="main">
    <nav>
      ...
    </nav>
    <section>
      <article>
        ...
      </article>
      <article>
        <figure>
          <img ... />
          <figcaption>...
        </figure>
        ...
      </article>
      <aside>
        ...
      </aside>
    </section>
    <section>
      ...
    </section>
  </div>
  <footer>
    ...
  </footer>
</body>
```

1

2

3

5

6

7

8

9

10

Header and Footer

<header> <footer>

Most web site pages have a recognizable header and footer section.

Typically the **header** contains

- the site logo
- title (and perhaps additional subtitles or taglines)
- horizontal navigation links, and
- perhaps one or two horizontal banners.

Header and Footer

<header> <footer>

The typical footer contains less important material, such as

- smaller text versions of the navigation,
- copyright notices,
- information about the site’s privacy policy, and
- perhaps twitter feeds or links to other social sites.

Header and Footer

Both the HTML5 <header> and <footer> element can be used not only for *page* headers and footers, they can also be used for header and footer elements within other HTML5 containers, such as <article> or <section>.

```
<header>
   <img src="logo.gif" alt="logo" />
   <h1>Introduction to HTML5</h1>
   ...
</header>
<article>
   <header>
       <h2>HTML5 Semantic Structure Elements </h2>
       <p>By <em>Mike Fennelly</em></p>
       <p><time>September 30, 2012</time></p>
   </header>
   ...
</article>
```

Heading Groups

The <hgroup> element can be used to group related headings together within one container.

```
<header>
   <hgroup>
       <h1>Chapter Two: HTML 1</h1>
     <h2>An Introduction</h2>
   </hgroup>
</header>
<article>
   <hgroup>
      <h2>HTML5 Semantic Structure Elements </h2>
    <h3>Overview</h3>
   </hgroup>
</article>
```

Navigation

<nav>

The **<nav>** element represents a section of a page that contains links to other pages or to other parts within the same page.

Like the other new HTML5 semantic elements, the browser does not apply any special presentation to the <nav> element.

The <nav> element was intended to be used for major navigation blocks, presumably the global and secondary navigation systems.

Navigation

```
<header>
   <img src="logo.gif" alt="logo" />
   <h1>Fundamentals of Web Development</h1>
   <nav role="navigation">
      <ul>
         <li><a href="index.html">Home</a></li>
         <li><a href="about.html">About Us</a></li>
         <li><a href="browse.html">Browse</a></li>
      </ul>
   </nav>
</header>
```

Articles and Sections

The **<article>** element represents a section of content that forms an independent part of a document or site; for example, a magazine or newspaper article, or a blog entry.

The **<section>** element represents a section of a document, typically with a title or heading.

Articles and Sections

According to the W3C, **<section>** is a much broader element, while the **<article>** element is to be used for blocks of content that could potentially be read or consumed independently of the other content on the page.

Sections versus Divs

How to decide which to use

<section> element is **not** a generic container element. HTML already has the element for such uses.

When an element is needed only for styling purposes or as a convenience for scripting, it makes sense to use the element instead.

Another way to help you decide whether or not to use the <section> element is to ask yourself if it is appropriate for the element's contents to be listed explicitly in the document's outline.

If so, then use a <section>; otherwise use a .

Figure and Figure Captions

<figure> <figcaption>

The W3C Recommendation indicates that the <figure> element can be used not just for images but for any type of *essential* content that could be moved to a different location in the page or document and the rest of the document would still make sense.

Figure and Figure Captions

Note however …

The **<figure>** element should **not** be used to wrap every image.

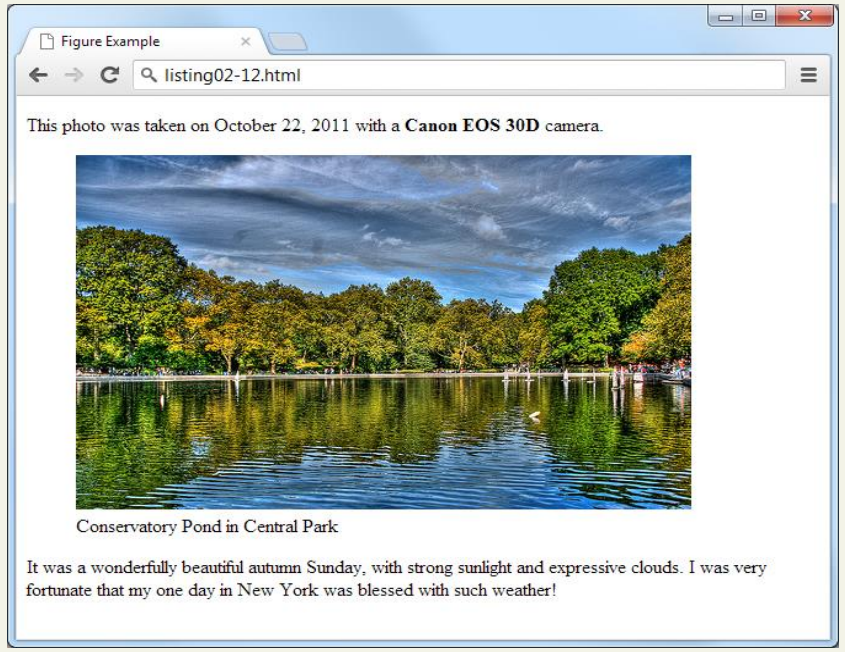
For instance, it makes no sense to wrap the site logo or non-essential images such as banner ads and graphical embellishments within <figure> elements.

Instead, only use the <figure> element for circumstances where the image (or other content) has a caption and where the figure is essential to the content but its position on the page is relatively unimportant.

Figure and Figure Captions

Figure could be moved to a different location in document …
But it has to exist in the document (i.e., the figure isn't optional)

```
<p>This photo was taken on October 22, 2011 with a Canon EOS 30D camera.</p>
<figure>
  <img src="images/central-park.jpg" alt="Central Park" /><br/>
  <figcaption>Conservatory Pond in Central Park</figcaption>
</figure>
<p>
It was a wonderfully beautiful autumn Sunday, with strong sunlight and
expressive clouds. I was very fortunate that my one day in New York was
blessed with such weather!
</p>
```



Aside

<aside>

The **<aside>** element is similar to the **<figure>** element in that it is used for marking up content that is separate from the main content on the page.

But while the **<figure>** element was used to indicate important information whose location on the page is somewhat unimportant, the **<aside>** element “represents a section of a page that consists of content that is tangentially related to the content around the aside element.”

The **<aside>** element could thus be used for sidebars, pull quotes, groups of advertising images, or any other grouping of non-essential elements.

Semantic Tags on tables

- <caption>
- <col>,<colgroup>
- <thead>
- <tfoot>
- <tbody>

A title for the table is good for accessibility.

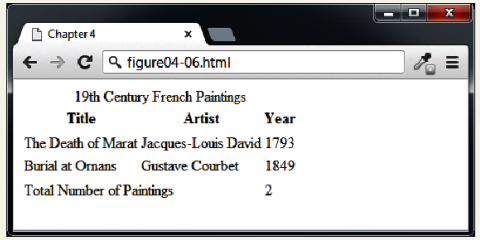
These describe our columns, and can be used to aid in styling.

Table header could potentially also include other <tr> elements.

Yes, the table footer comes *before* the body.

Potentially, with styling the browser can scroll this information, while keeping the header and footer fixed in place.

```
<table>
  <caption>19th Century French Paintings</caption>
  <col class="artistName" />
  <colgroup id="paintingColumns">
      <col />
      <col />
  </colgroup>

  <thead>
     <tr>
        <th>Title</th>
        <th>Artist</th>
        <th>Year</th>
     </tr>
  </thead>

  <tfoot>
     <tr>
        <td colspan="2">Total Number of Paintings</td>
        <td>2</td>
     </tr>
  </tfoot>

  <tbody>
     <tr>
        <td>The Death of Marat</td>
        <td>Jacques-Louis David</td>
        <td>1793</td>
     </tr>
     <tr>
        <td>Burial at Ornans</td>
        <td>Gustave Courbet</td>
        <td>1849</td>
     </tr>
  </tbody>

</table>
```



INTRODUCING FORMS

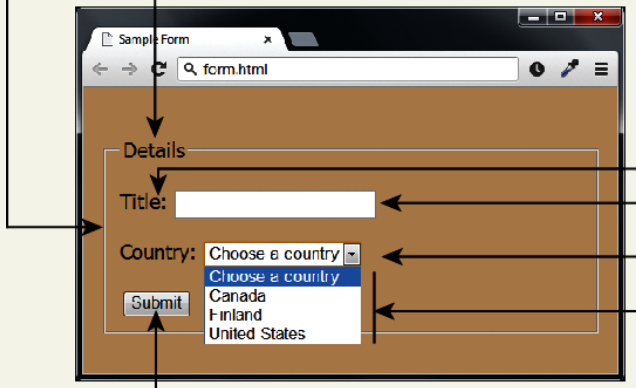
HTML Forms

Richer way to interact with server

Forms provide the user with an alternative way to interact with a web server.

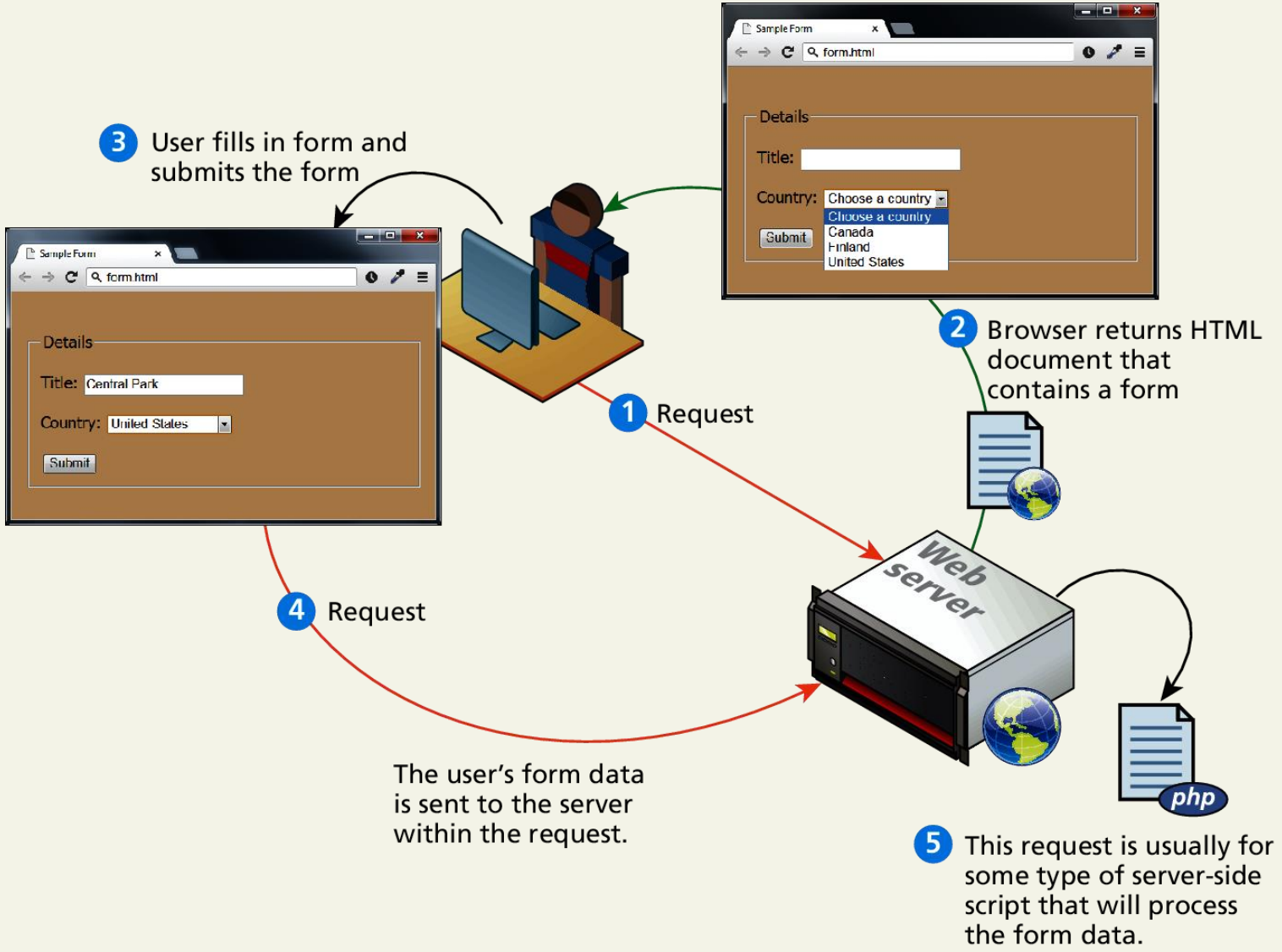
- Forms provide rich mechanisms like:
  - Text input
  - Password input
  - Options Lists
  - Radio and check boxes

Form Structure




```
<form method="get" action="process.php">
  <fieldset>
    <legend>Details</legend>
    <p>
      <label>Title: </label>
      <input type="text" name="title" />
    </p>
    <p>
      <label>Country: </label>
      <select name="where">
        <option>Choose a country</option>
        <option>Canada</option>
        <option>Finland</option>
        <option>United States</option>
      </select>
    </p>
    <input type="submit" />
  </fieldset>
</form>
```

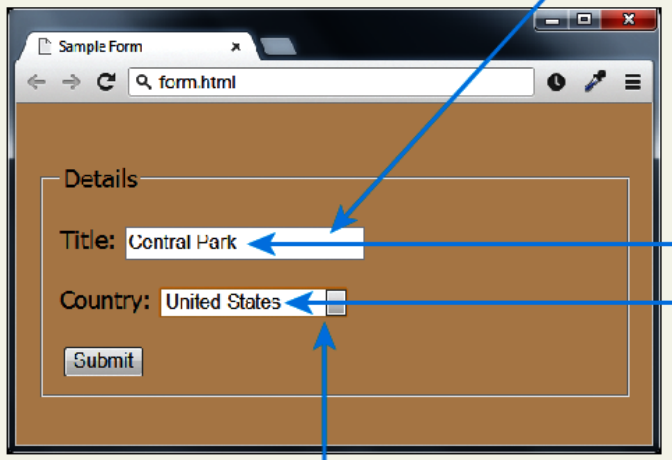
How forms interact with servers



Query Strings

At the end of the day, another string

```
<input type="text" name="title" />
```

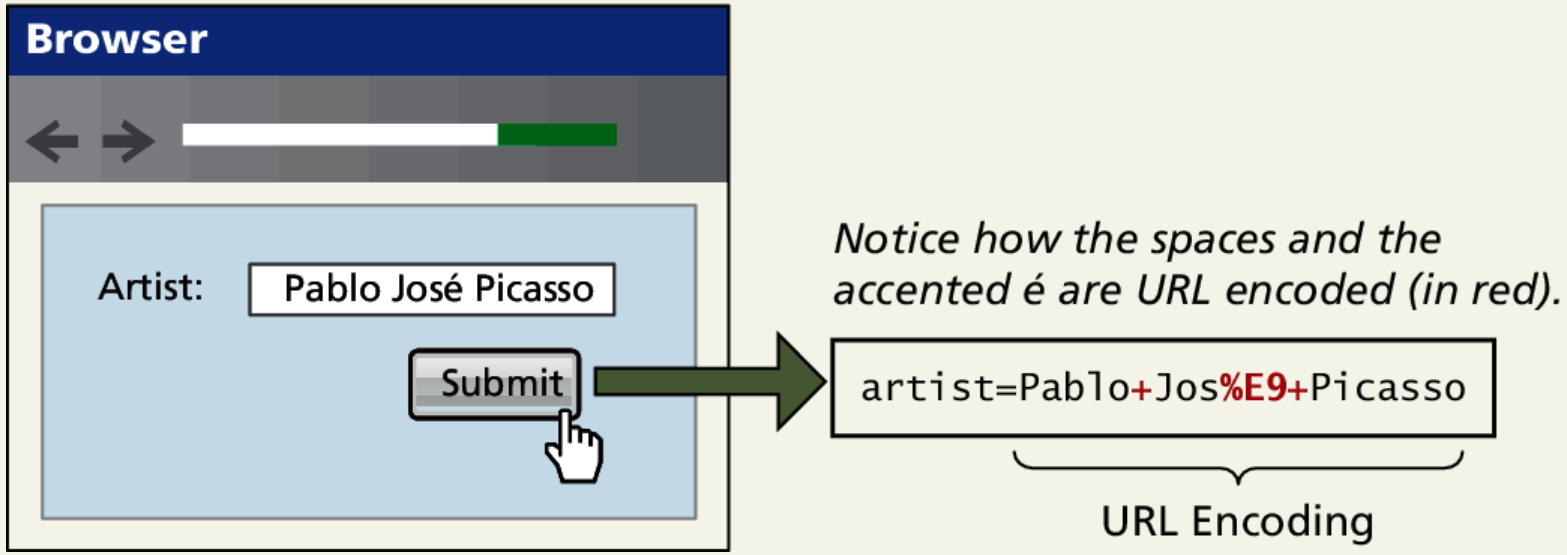



```
title=Central+Park&where=United+States
```

```
<select name="where">
```

URL encoding

Special symbols



<form> element

Two essential features of any form, namely the **action** and the **method** attributes.

- The **action** attribute specifies the URL of the server-side resource that will process the form data
- The **method** attribute specifies how the query string data will be transmitted from the browser to the server.
  - GET
  - POST

GET vs POST


```
<form method="get" action="process.php">
```

```
GET /process.php?title=Central+Park&where=United+States http/1.1
```

querystring

```
<form method="post" action="process.php">
```

```
POST /process.php http/1.1
Date: Sun, 20 May 2012 23:59:59 GMT
Host: www.mysite.com
User-Agent: Mozilla/4.0
Content-Length: 47

title=Central+Park&where=United+States
```

HTTP Header

querystring

GET vs POST

Advantages and Disadvantages

GET

- Data can be clearly seen in the address bar.
- Data remains in browser history and cache.
- Data can be bookmarked
- Limit on the number of characters in the form data returned.

POST

- Data can contain binary data.
- Data is hidden from user.
- Submitted data is not stored in cache, history, or bookmarks.

Section 4 of 6

FORMS CONTROL ELEMENTS

Form-Related HTML Elements

Type	Description
<button>	Defines a clickable button.
<datalist>	An HTML5 element form defines lists to be used with other form elements.
<fieldset>	Groups related elements in a form together.
<form>	Defines the form container.
<input>	Defines an input field. HTML5 defines over 20 different types of input.
<label>	Defines a label for a form input element.
<legend>	Defines the label for a fieldset group.
<option>	Defines an option in a multi-item list.
<optgroup>	Defines a group of related options in a multi-item list.
<select>	Defines a multi-item list.
<textarea>	Defines a multiline text entry box.

Text Input Controls

Type	Description
text	Creates a single line text entry box. **<input type="text" name="title" />**
textarea	Creates a multiline text entry box. **<textarea rows="3" ... />**
password	Creates a single line text entry box for a password **<input type="password" ... />**
search	Creates a single-line text entry box suitable for a search string. This is an HTML5 element. **<input type="search" … />**
email	Creates a single-line text entry box suitable for entering an email address. This is an HTML5 element. **<input type="email" … />**
tel	Creates a single-line text entry box suitable for entering a telephone. This is an HTML5 element. **<input type="tel" … />**
url	Creates a single-line text entry box suitable for entering a URL. This is an HTML5 element. **<input type="url" … />**

Text Input Controls

Classic

```
<input type="text" ... />
```

Text:

```
<textarea>
  enter some text
</textarea>
```

```
<textarea placeholder="enter some text">
</textarea>
```

TextArea: enter some text

TextArea: Enter some text

```
<input type="password" ... />
```

Password:

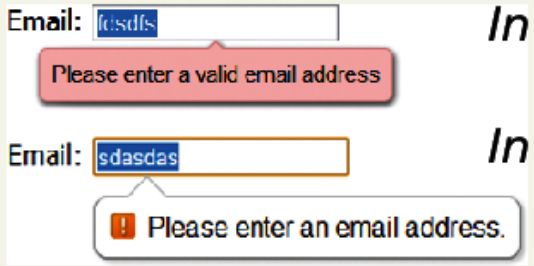
Password: ••••

Text Input Controls

HTML5

```
<input type="search" placeholder="enter search text" ... />
```


```
<input type="email" ... />
```



In Opera

In Chrome

```
<input type="url" ... />
```


```
<input type="tel" ... />
```

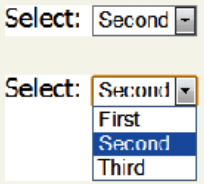
Select Lists

Chose an option, any option.

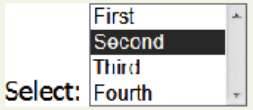
- **<select>** element is used to create a multiline box for selecting one or more items
  - The options are defined using the **<option>** element
  - can be hidden in a dropdown or multiple rows of the list can be visible
  - Option items can be grouped together via the **<optgroup>** element.

Select Lists

Select List Examples




```
<select name="choices">
  <option>First</option>
  <option selected>Second</option>
  <option>Third</option>
</select>
```




```
<select size="3" ... >
```


```
<select … >
 <optgroup label="North America">
   <option>Calgary</option>
   <option>Los Angeles</option>
 </optgroup>
 <optgroup label="Europe">
   <option>London</option>
   <option>Paris</option>
   <option>Prague</option>
 </optgroup>
</select>
```

Which Value to send

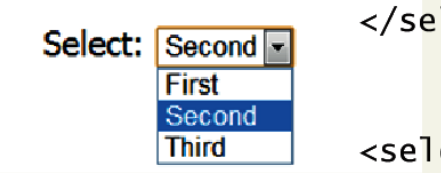
Select Lists Cont.

The **value** attribute of the <option> element is used to specify what value will be sent back to the server.

The value attribute is optional; if it is not specified, then the text within the container is sent instead

```
<select name="choices">
  <option>First</option>
  <option>Second</option>
  <option>Third</option>
</select>
```

?choices=Second




```
<select name="choices">
  <option value="1">First</option>
  <option value="2">Second</option>
  <option value="3">Third</option>
</select>
```

?choices=2

Radio Buttons

Radio buttons are useful when you want the user to select a single item from a small list of choices and you want all the choices to be visible

- radio buttons are added via the **<input type="radio">** element
- The buttons are mutually exclusive (i.e., only one can be chosen) by sharing the same name attribute
- The checked attribute is used to indicate the default choice
- the value attribute works in the same manner as with the <option> element

Radio Buttons

Continent:

- North America
- South America
- Asia

```
<input type="radio" name="where" value="1">North America<br/>
<input type="radio" name="where" value="2" checked>South America<br/>
<input type="radio" name="where" value="3">Asia
```

Checkboxes

Checkboxes are used for getting yes/no or on/off responses from the user.

- checkboxes are added via **the <input type="checkbox”>** Element
- You can also group checkboxes together by having them share the same name attribute
- Each checked checkbox will have its value sent to the server
- Like with radio buttons, the checked attribute can be used to set the default value of a checkbox

Checkboxes

I accept the software license ☑

Where would you like to visit?
☑ Canada
☐ France
☑ Germany

```
<label>I accept the software license</label>
<input type="checkbox" name="accept" >
```

```
<label>Where would you like to visit? </label><br/>
<input type="checkbox" name="visit" value="canada">Canada<br/>
<input type="checkbox" name="visit" value="france">France<br/>
<input type="checkbox" name="visit" value="germany">Germany
```

```
?accept=on&visit=canada&visit=germany
```

Button Controls

Type	Description
<input type="submit">	Creates a button that submits the form data to the server.
<input type="reset">	Creates a button that clears any of the user's already entered form data.
<input type="button">	Creates a custom button. This button may require Javascript for it to actually perform any action.
<input type="image">	Creates a custom submit button that uses an image for its display.
<button>	Creates a custom button. The <button> element differs from <input type="button"> in that you can completely customize what appears in the button; using it, you can, for instance, include both images and text, or skip server-side processing entirely by using hyperlinks. You can turn the button into a submit button by using the type="submit" attribute.

Button Controls

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


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

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

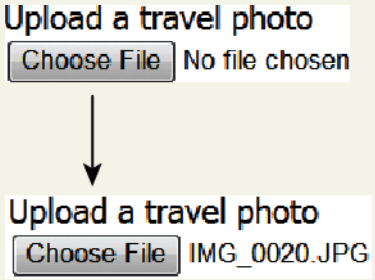

```
<input type="image" src="appointment.png" />
```


```
<button>
   <a href="email.html">
      <img src="images/email.png" alt=""/>
      Email
   </a>
</button>
```

```
<button type="submit"  >
   <img src="images/edit.png" alt=""/>
   Edit
</button>
```

Specialized Controls

- <input type=hidden>
- <input type=file>



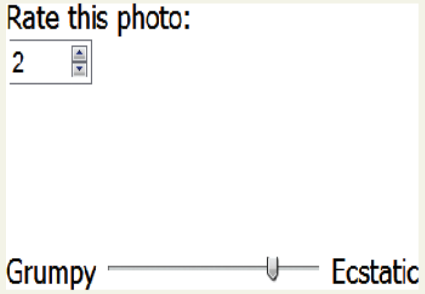

```
<form method="post" enctype="multipart/form-data" ... >
   ...
   <label>Upload a travel photo</label>
   <input type="file" name="photo" />
   …
</form>
```

Number and Range

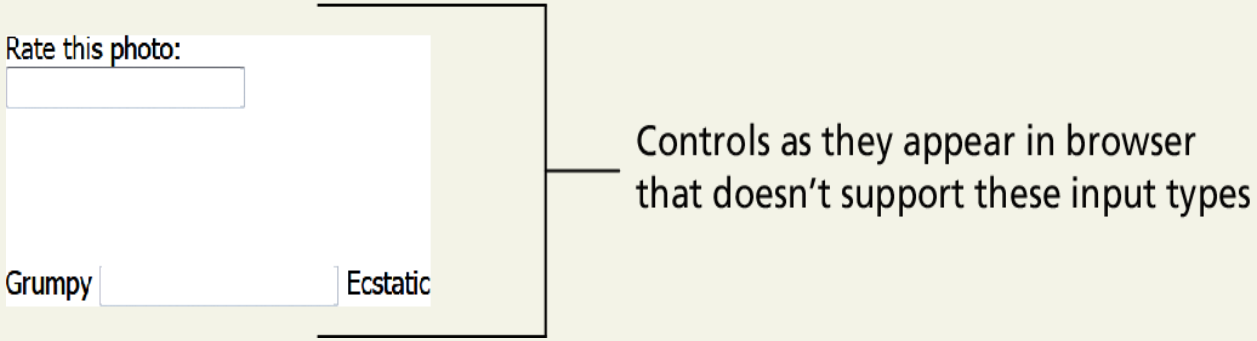
Typically input values need be **validated**. Although server side validation is required, optional client side pre-validation is good practice.

The number and range controls Added in HTML5 provide a way to input numeric values that **eliminates the need for JavaScript numeric validation!!!**

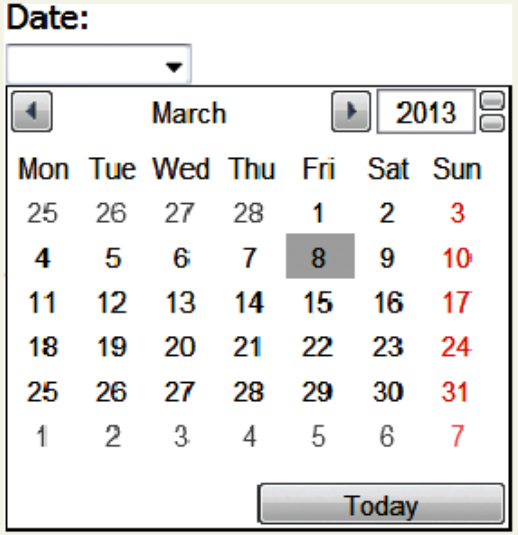
Number and Range



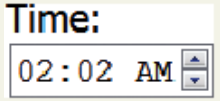

```
<label>Rate this photo: <br/>
<input type="number" min="1" max="5" name="rate" />


Grumpy
<input type="range" min="0" max="10" step="1" name="happiness" />
Ecstatic
```



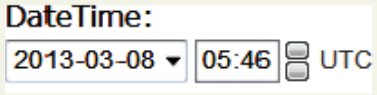
HTML5 Date and Time Controls



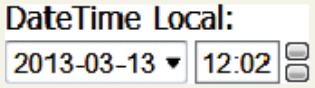

```
<label>Date: <br/>
<input type="date" ... />
```




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

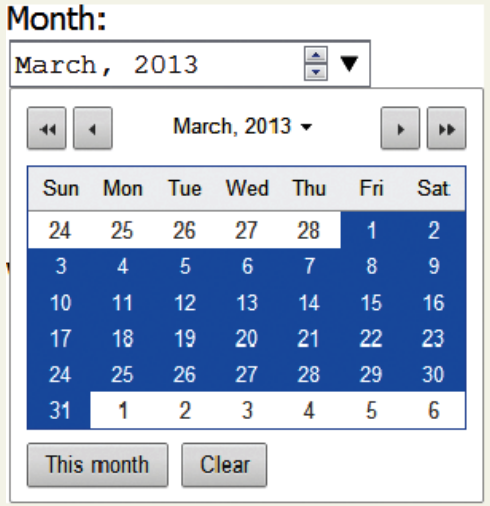



```
<input type="datetime" ... />
```

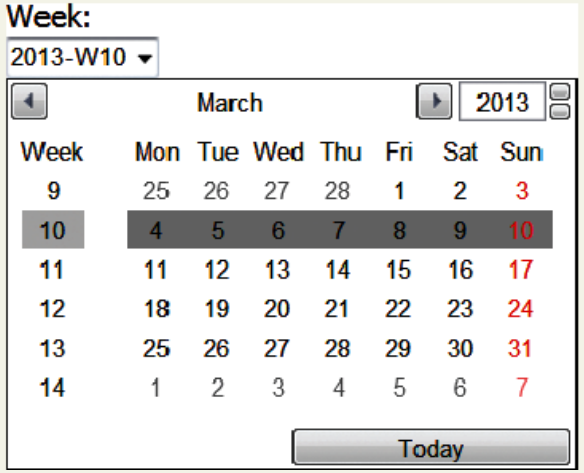



```
<input type="datetime-local" ... />
```

HTML5 Date and Time Controls




```
<input type="month" ... />
```




```
<input type="week" ... />
```

HTML Controls

Type	Description
date	Creates a general date input control. The format for the date is "yyyy-mm-dd".
time	Creates a time input control. The format for the time is "HH:MM:SS", for hours:minutes:seconds.
datetime	Creates a control in which the user can enter a date and time.
datetime-local	Creates a control in which the user can enter a date and time without specifying a time zone.
month	Creates a control in which the user can enter a month in a year. The format is "yyyy-mm".
week	Creates a control in which the user can specify a week in a year. The format is "yyyy-W##".