2. Introduction to HTML

2023학년 2학기 웹응용프로그래밍

권 동 현

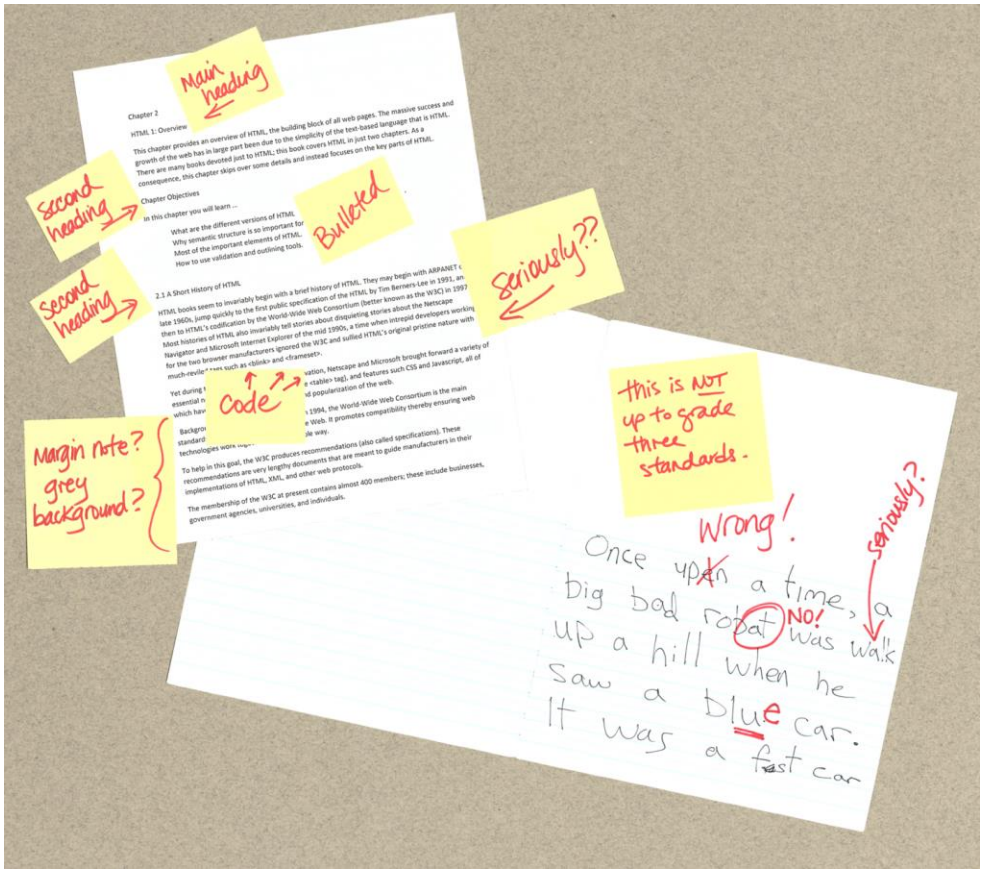
CSLAB

Contents

HTML Syntax

HTML

- 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.
  - The term comes from the days of print, when editors would write instructions on manuscript pages that might be revision instructions to the author or copy editor.



HTML tags

- At its simplest, **markup** is a way to indicate information about the content
  - This "information about content" in HTML is implemented via **tags** (aka elements).
  - 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.

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

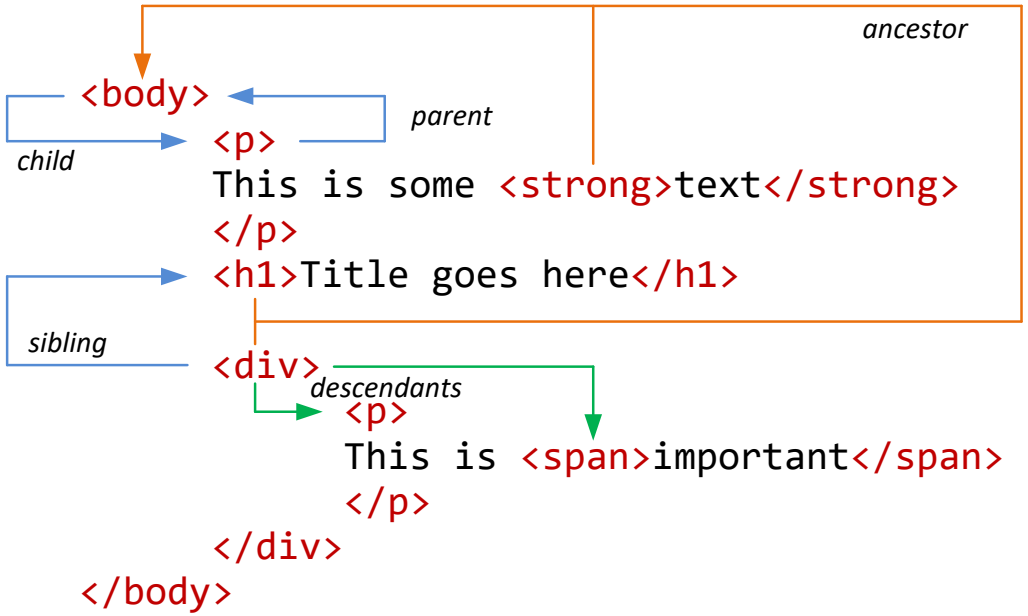
Elements and Attributes

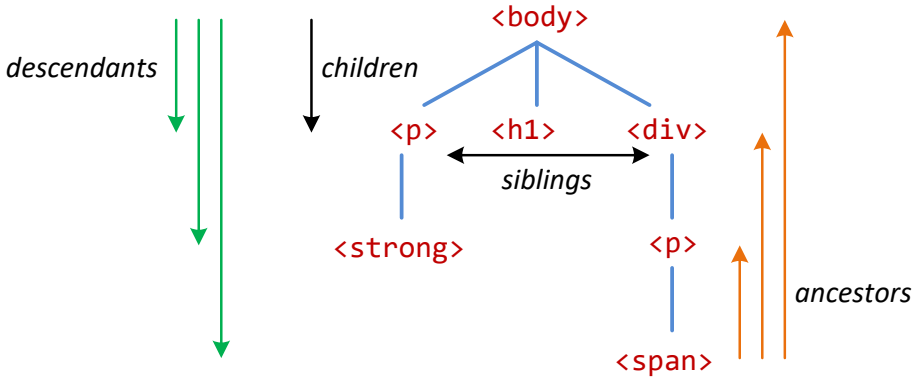
- **HTML documents** are composed of textual content and HTML elements.
- An **HTML element** can contain text, other elements, or be empty. It is identified in the HTML document by tags.
- HTML elements can also contain attributes. An **HTML attribute** is a name=value pair that provides more information about the HTML element.

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**.

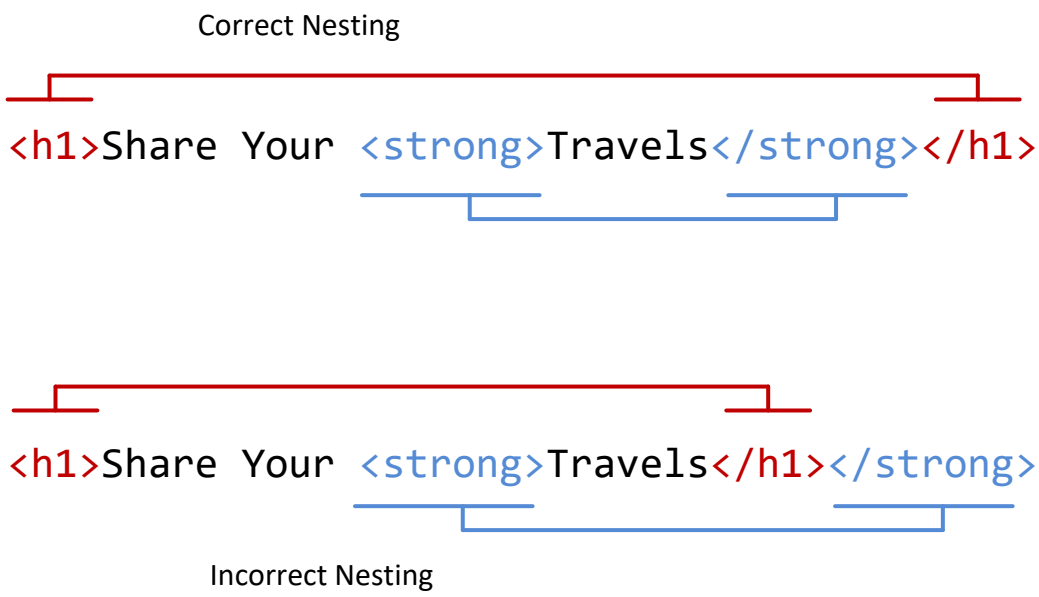
Nesting HTML Elements





Nesting HTML elements

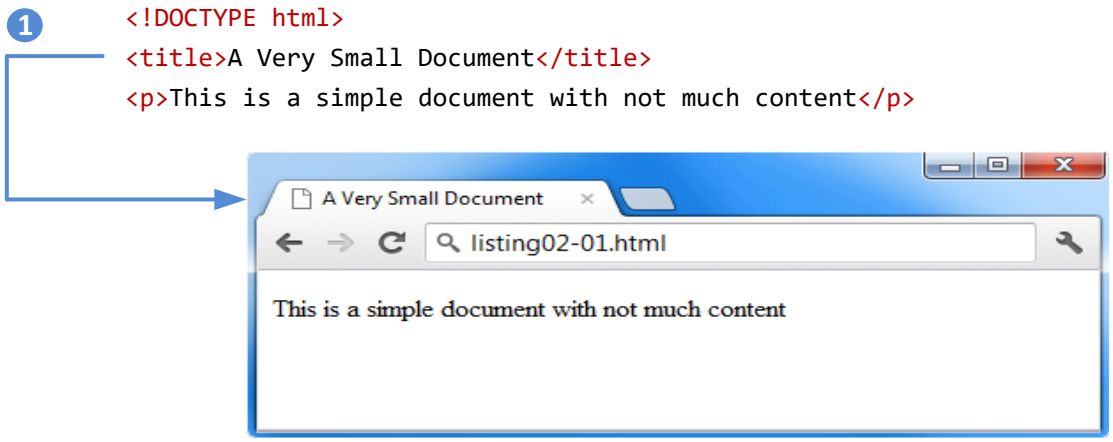
- In order to properly construct a hierarchy of elements, your browser expects each HTML nested element to be properly nested.
- That is, a child's ending tag must occur before its parent's ending tag.



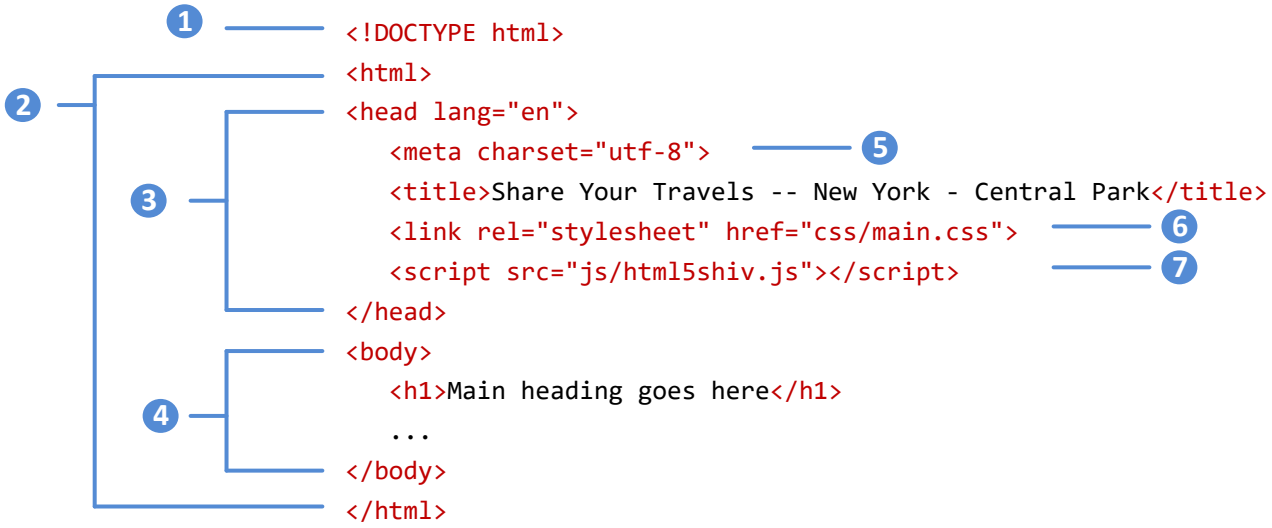
Structure of HTML

Simplest HTML document

- 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.

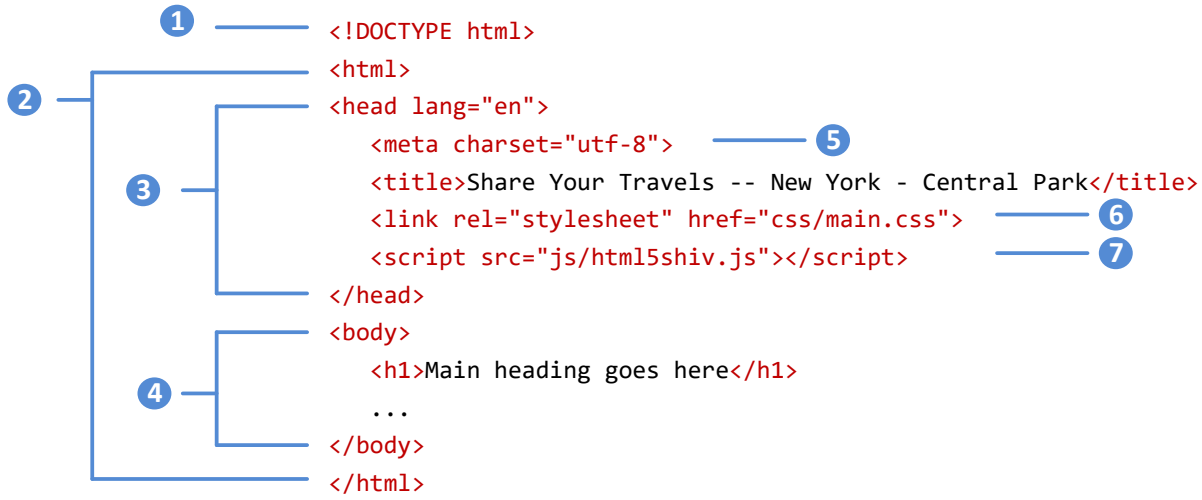


A more complete document



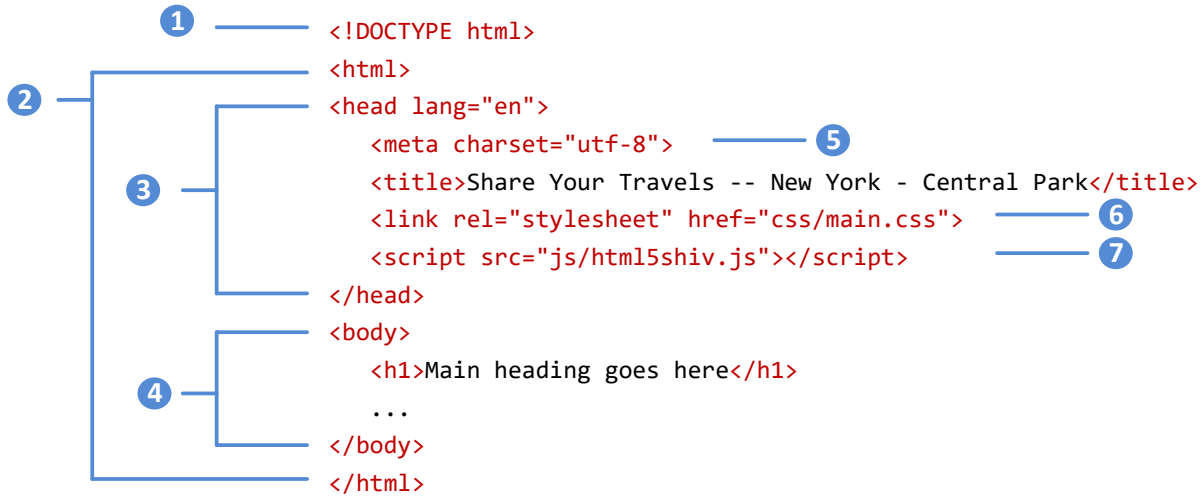
1 DOCTYPE

- 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.



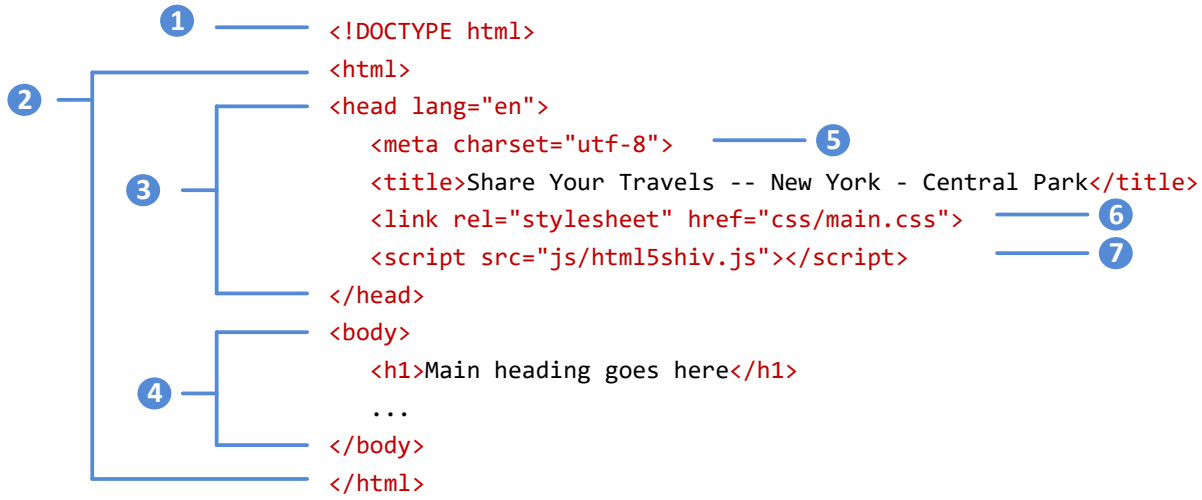
HTML, Head, and Body

- HTML5 does not require the use of the <html>, <head>, and <body>.
- However, in XHTML they were required, and most web authors continue to use them.
- The <html> element is sometimes called the root element as it contains all the other HTML elements in the document.



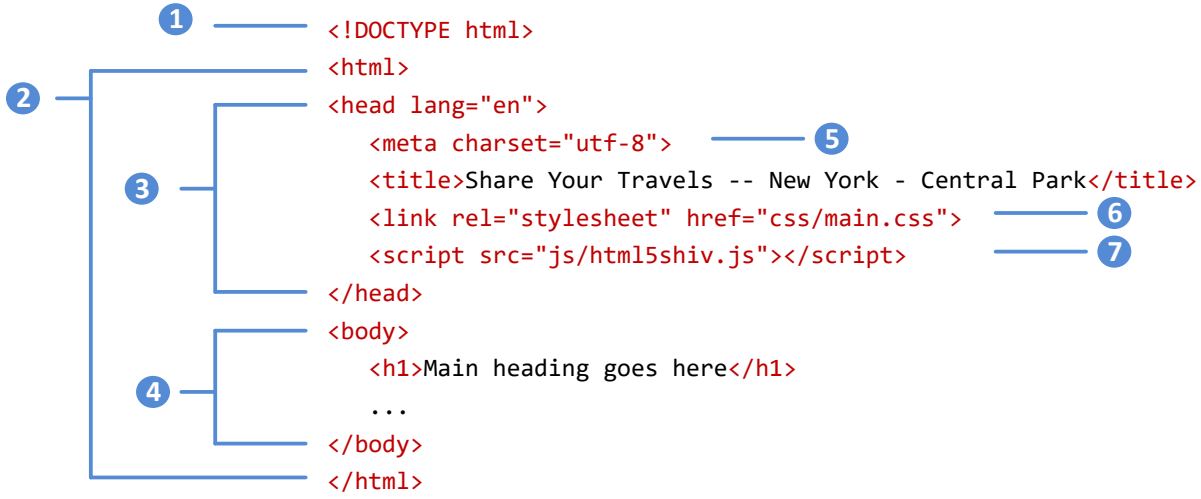
Head and Body

- HTML pages are divided into two sections: the **head** and the **body**, which correspond to the <head> and <body> elements.
- The head contains descriptive elements about the document
- The body contains content that will be displayed by the browser.



Inside the head

- 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.
- 6 Our example specifies an external CSS style sheet file that is used with this document.
- 7 It also references an external Javascript file.



Quick Tour of HTML

Why a quick tour?

- HTML5 contains many structural and presentation elements – too many to completely cover in this presentation.
- Rather than comprehensively cover all these elements, this presentation will provide a quick overview of the most common elements.

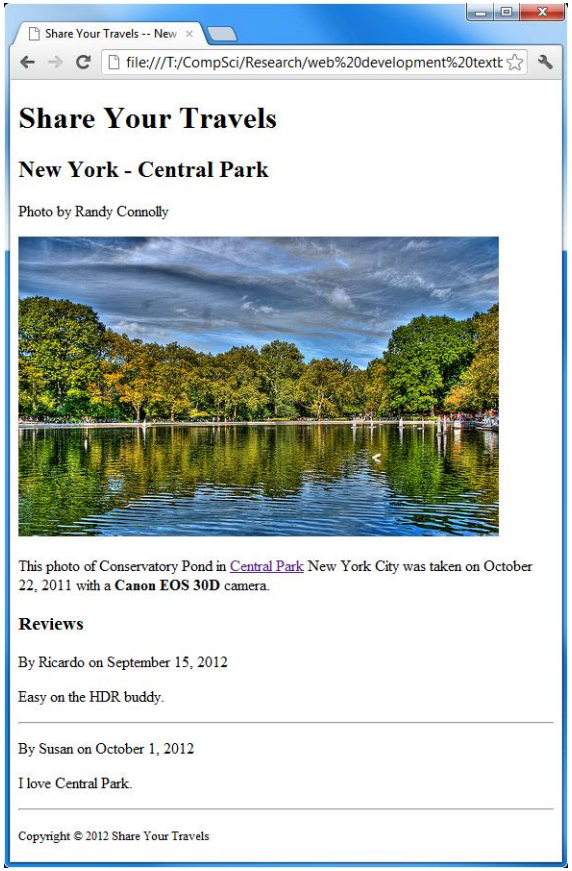
Sample Document

```
<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>

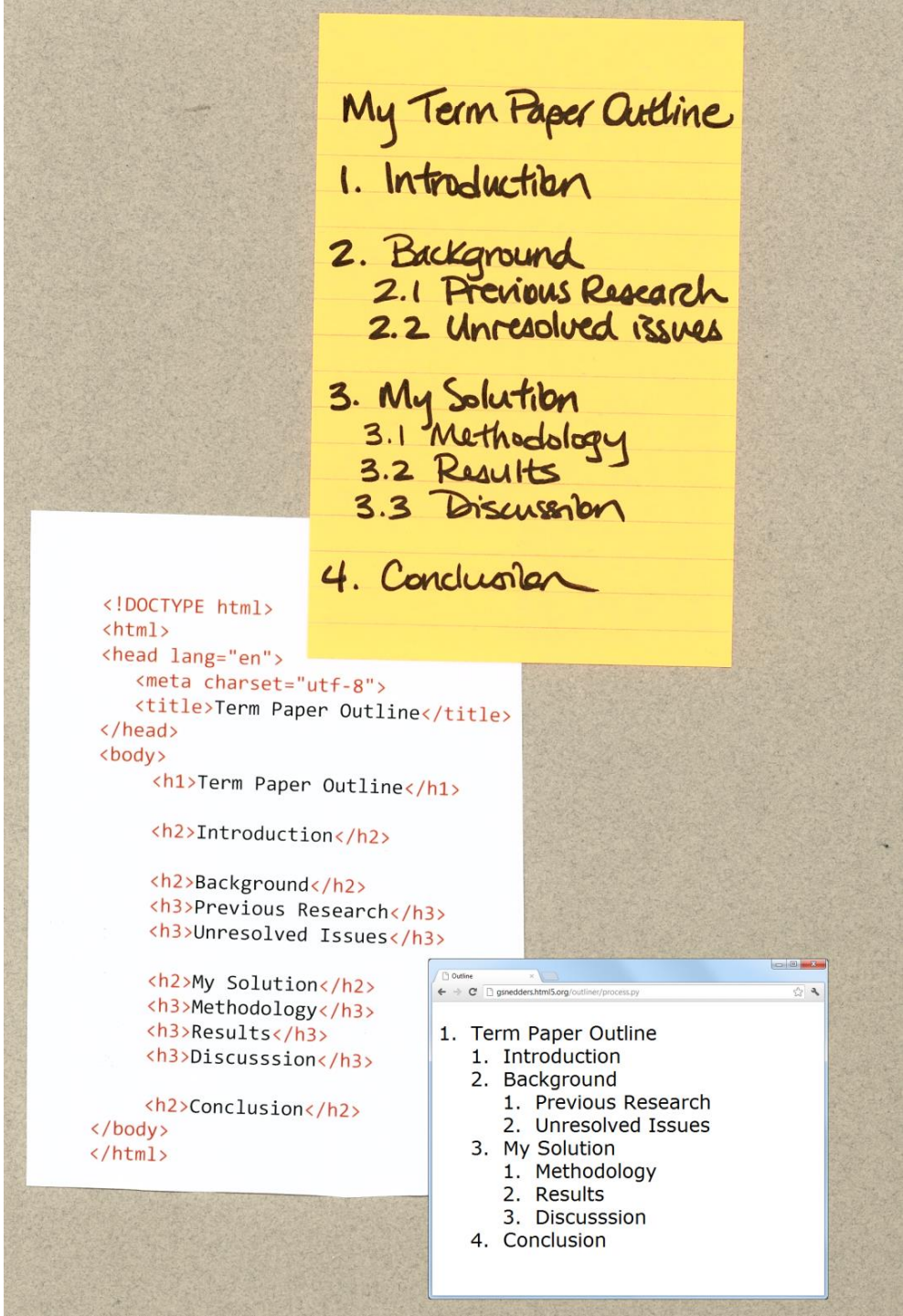
   <p><small>Copyright &copy; 2012 Share Your Travels</small></p>
</body>
```



1 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

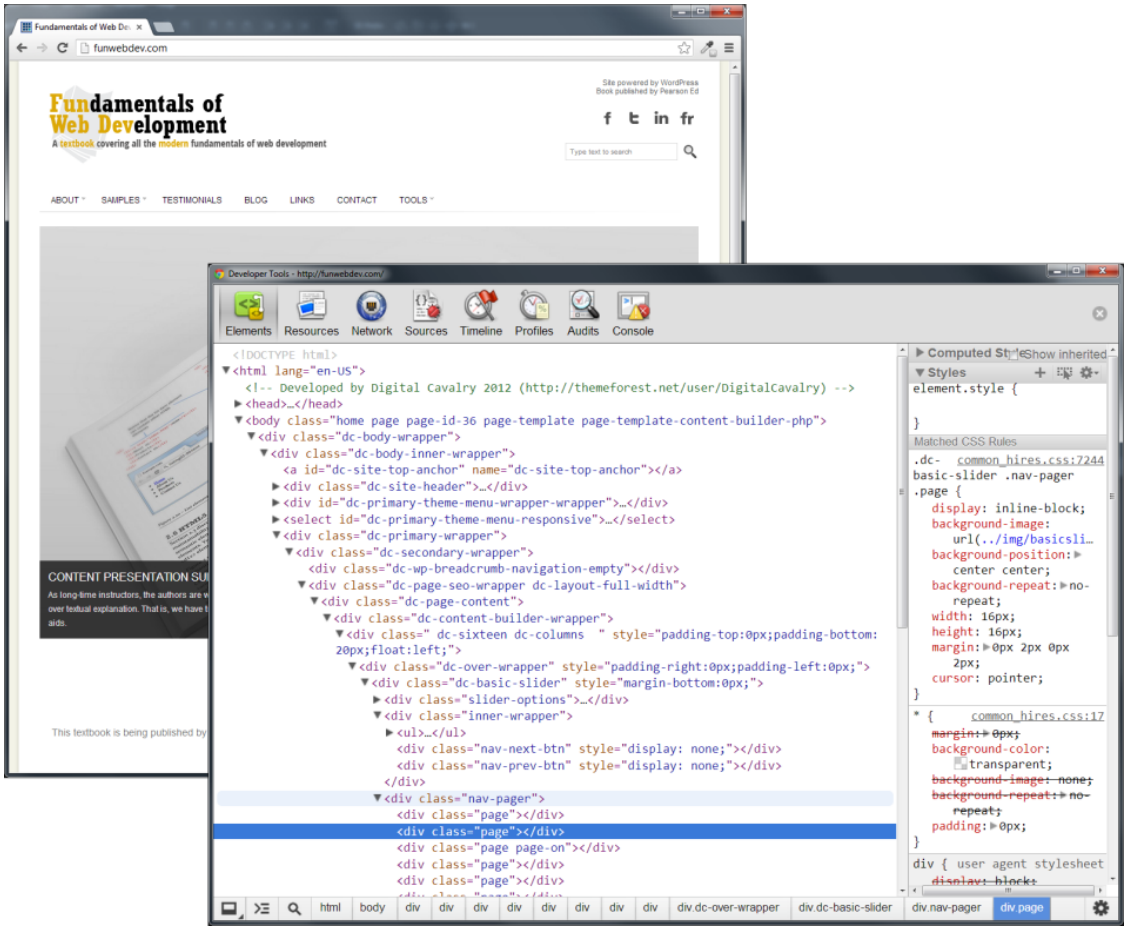
2 Paragraphs and 6 Divisions

- 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.

- 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.



3 Links

- 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

Link Text

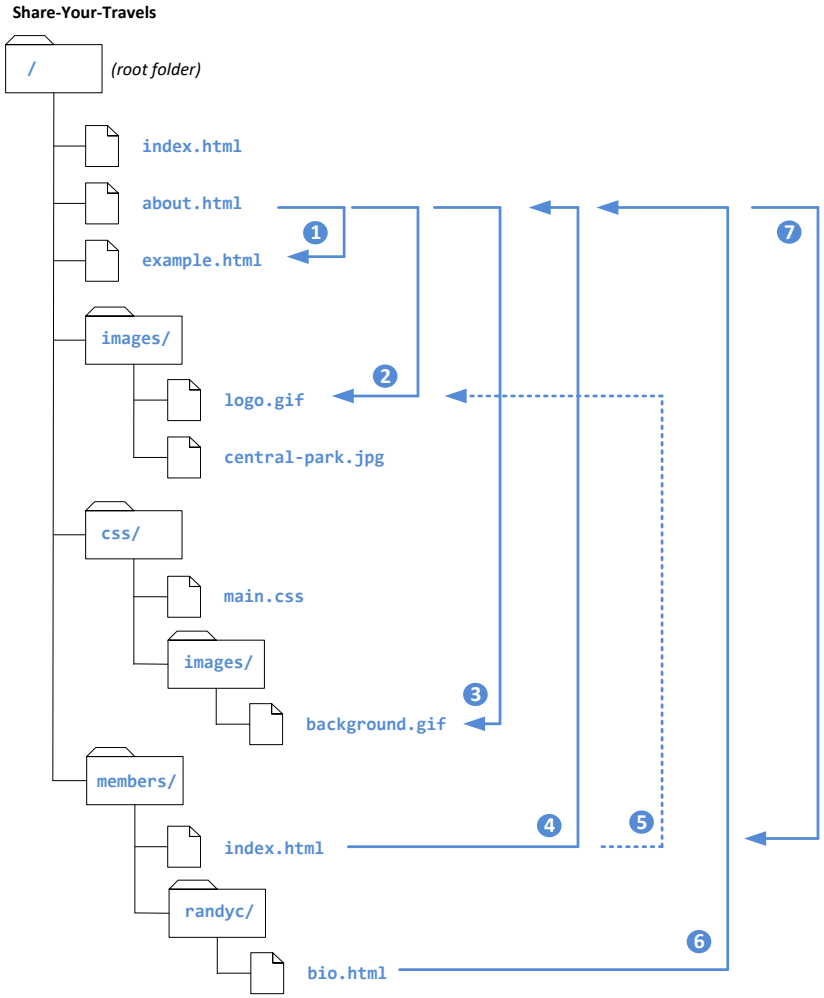
Some guidance . . . or . . . don't "Click Here"

- Links with the label “Click Here” were once a staple of the web.
- Today, such links are frowned upon, since:
  - they do not tell users where the link will take them
  - as a verb “click” is becoming increasingly inaccurate when one takes into account the growth of mobile browsers.
- Instead, textual link labels should be descriptive.
  - “Click here to see the race results”
  - “Race Results” or “See Race Results”.

URL Absolute Referencing

- URL Absolute Referencing
  - 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
  - We also need to be able to successfully reference files within our site.
  - This requires learning the syntax for so-called **relative referencing**.
  - When referencing a resource that is on the same server as your HTML document, then you can use briefer relative referencing. If the URL does not include the “http://” then the browser will request the current server for the file.
  - Pathnames on the web follow Unix conventions.
    - Forward slashes (“/”) are used to separate directory names from each other and from file names.
    - Double-periods (“..”) are used to reference a directory “above” the current one in the directory tree.

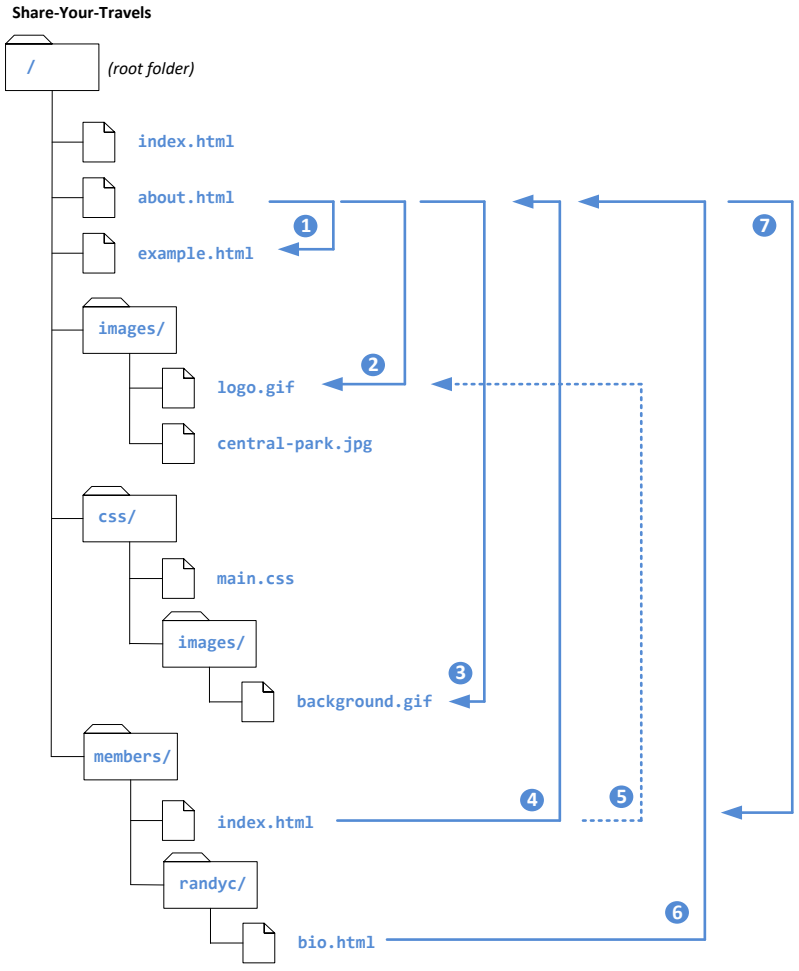
URL Relative Referencing



	Relative Link Type	Example
1	**Same Directory** To link to a file within the same folder, simply use the file name.	To link to example.html from about.html (in Figure 2.18), use: `<a href="example.html">`
2	**Child Directory** To link to a file within a subdirectory, use the name of the subdirectory and a slash before the file name.	To link to logo.gif from about.html, use: `<a href="images/logo.gif">`
3	**Grandchild/Descendant Directory** To link to a file that is multiple subdirectories *below* the current one, construct the full path by including each subdirectory name (separated by slashes) before the file name.	To link to background.gif from about.html, use: `<a href="css/images/background.gif">`

4	**Parent/Ancestor Directory**
	
	Use "../" to reference a folder *above* the current one. If trying to reference a file several levels above the current one, simply string together multiple "../".
		To link to about.html from index.html in members, use:
		
		`<a href="../about.html">`
		
		To link to about.html from bio.html, use:
		
		`<a href="../../about.html">`

URL Relative Referencing



5 Sibling Directory

Use "../"to move up to the appropriate level, and then use the same technique as for child or grandchild directories.

To link to logo.gif from index.html in members, use:

```
<a href="../images/about.html">
```

To link to background.gif from bio.html, use:

```
<a
href="../../css/images/background.gif">
```

6 Root Reference

An alternative approach for ancestor and sibling references is to use the so-called **root reference** approach. In this approach, begin the reference with the root reference (the "/") and then use the same technique as for child or grandchild directories. **Note that these will only work on the server! That is, they will not work when you test it out on your local machine.**

To link to about.html from bio.html, use:

```
<a href="/about.html">
```

To link to background.gif from bio.html, use:

```
<a href="/images/background.gif">
```

7 Default Document

Web servers allow references to directory names without file names. In such a case, the web server will serve the default document, which is usually a file called index.html (apache) or default.html (IIS). **Again, this will only generally work on the web server.**

To link to index.html in members from about.html, use either:

```
<a href="members">
```

Or

```
<a href="/members">
```

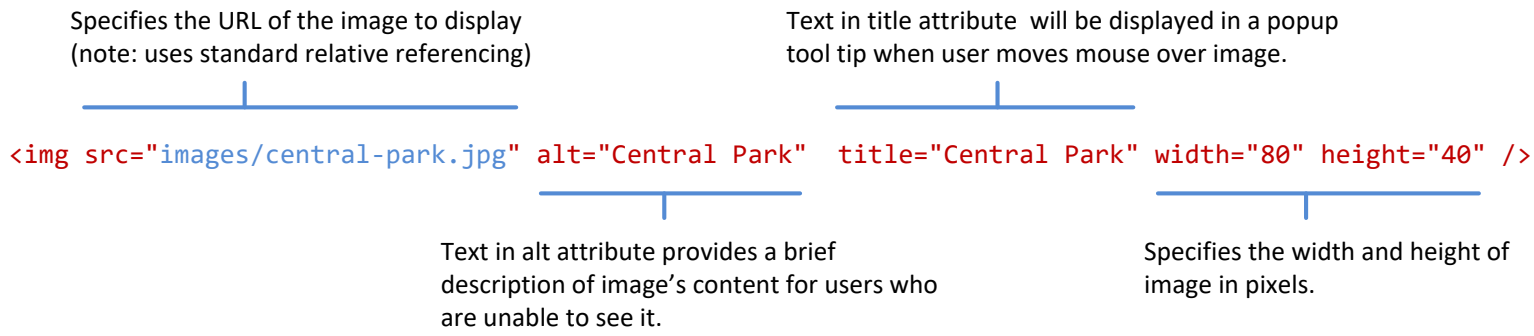
Inline Text Elements

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>,
, ,

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.



Lists

HTML provides three types of lists

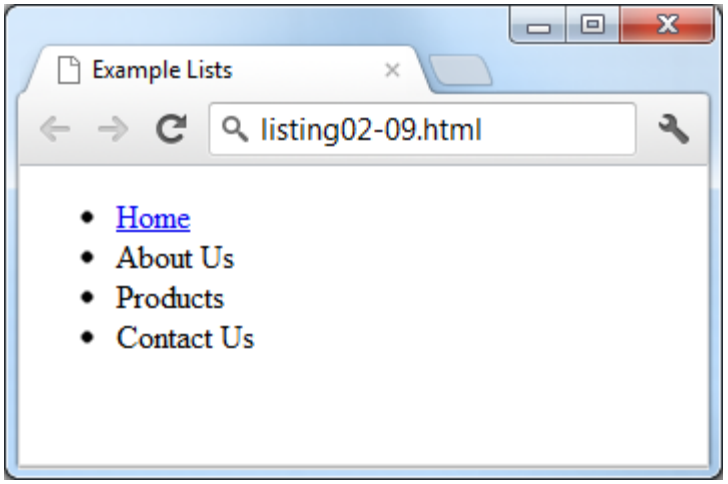
- **Unordered lists.** Collections of items in no particular order; these are by default rendered by the browser as a bulleted list.
- **Ordered lists.** Collections of items that have a set order; these are by default rendered by the browser as a numbered list.
- **Definition lists**. Collection of name and definition pairs. These tend to be used infrequently. Perhaps the most common example would be a FAQ list.

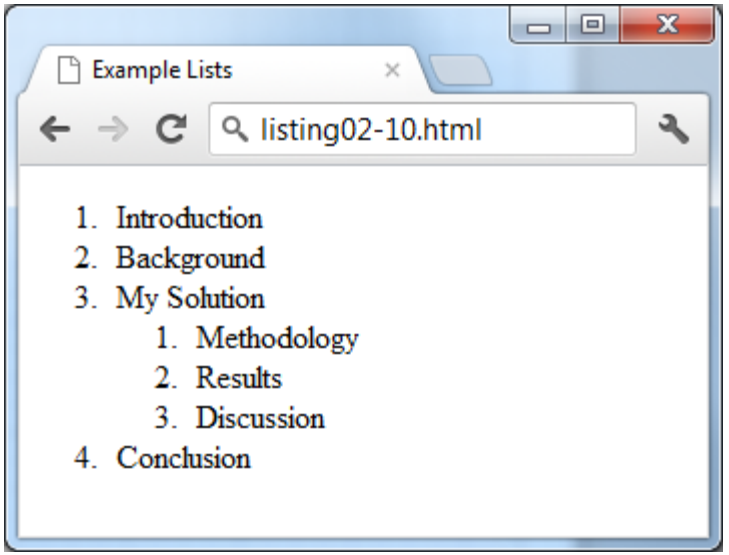
Lists

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>
```





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 ©

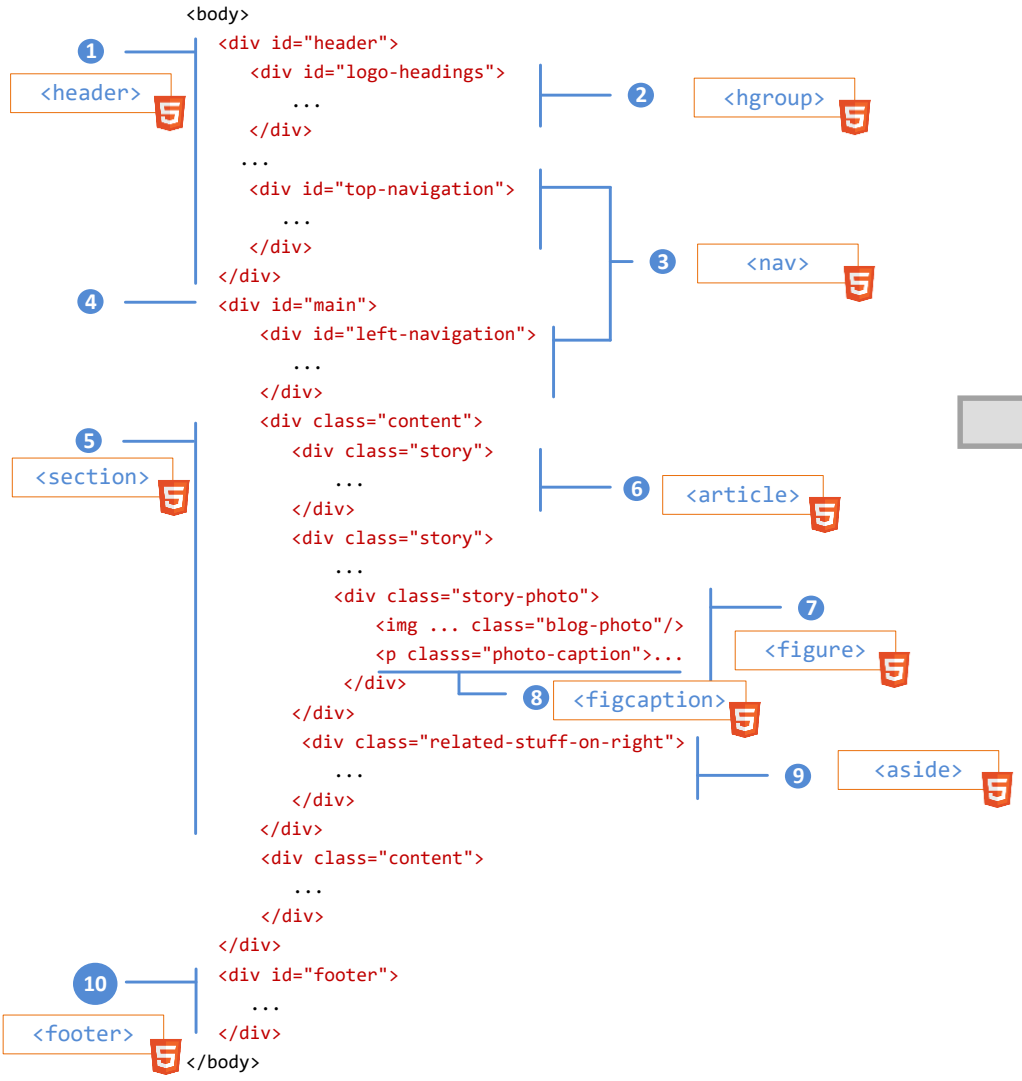
Semantic Elements

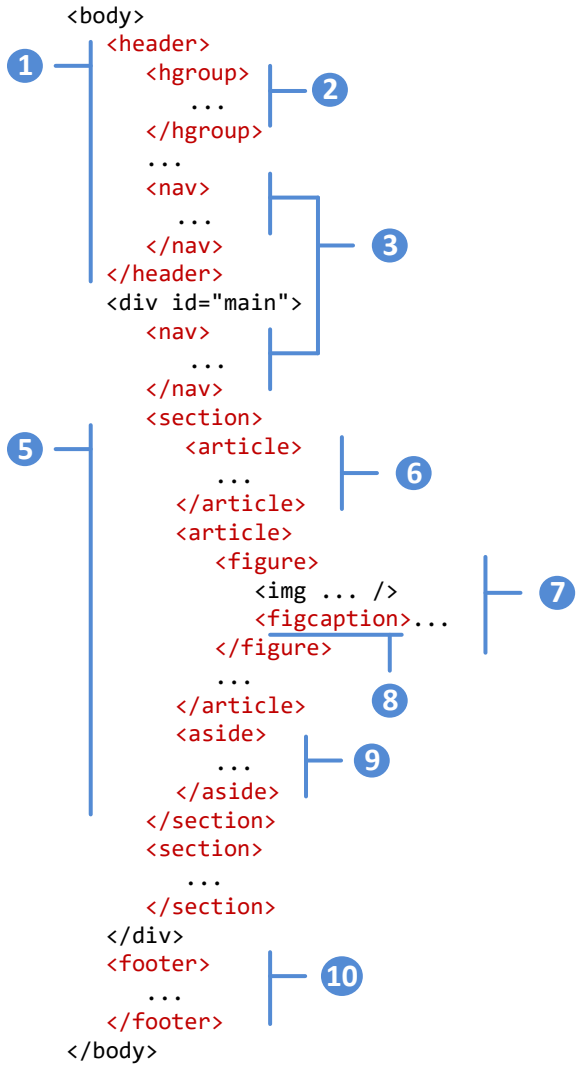
HTML5 Semantic Elements

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.

XHTML versus HTML5





1 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.
- 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>Fundamentals of Web Development</h1>
   ...
</header>
<article>
   <header>
       <h2>HTML5 Semantic Structure Elements </h2>
        <p>By <em>Randy Connolly</em></p>
        <p><time>September 30, 2012</time></p>
   </header>
   ...
</article>
```

❷ 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>
```

❸ Navigation

- 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.

```
<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>
```

❺ ❻ Articles and Sections

<article> <section>

- 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.
- 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.

- The WHATWG specification warns readers that the <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 .

❼ ❽ Figure and Figure Captions

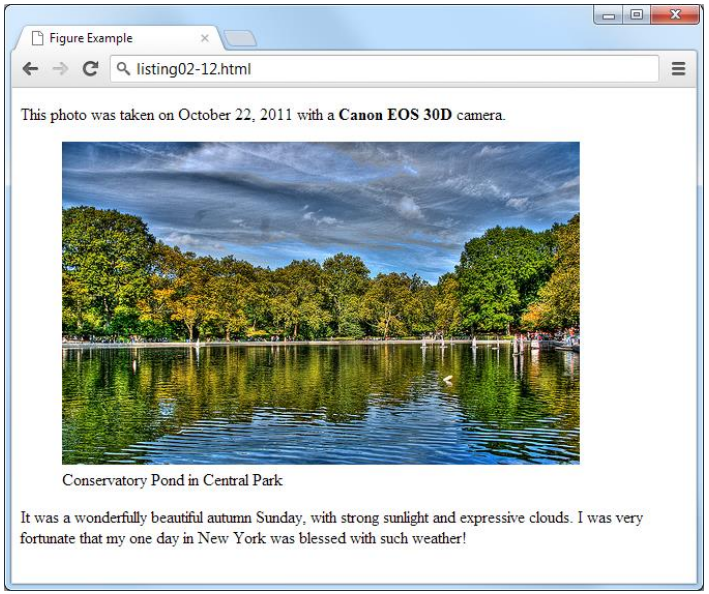
<figure> <figcaption>

- 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.
- 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 Markup

Semantic Markup

- 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).
- As a consequence, beginning HTML authors are often counseled 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.
  - 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

- 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.

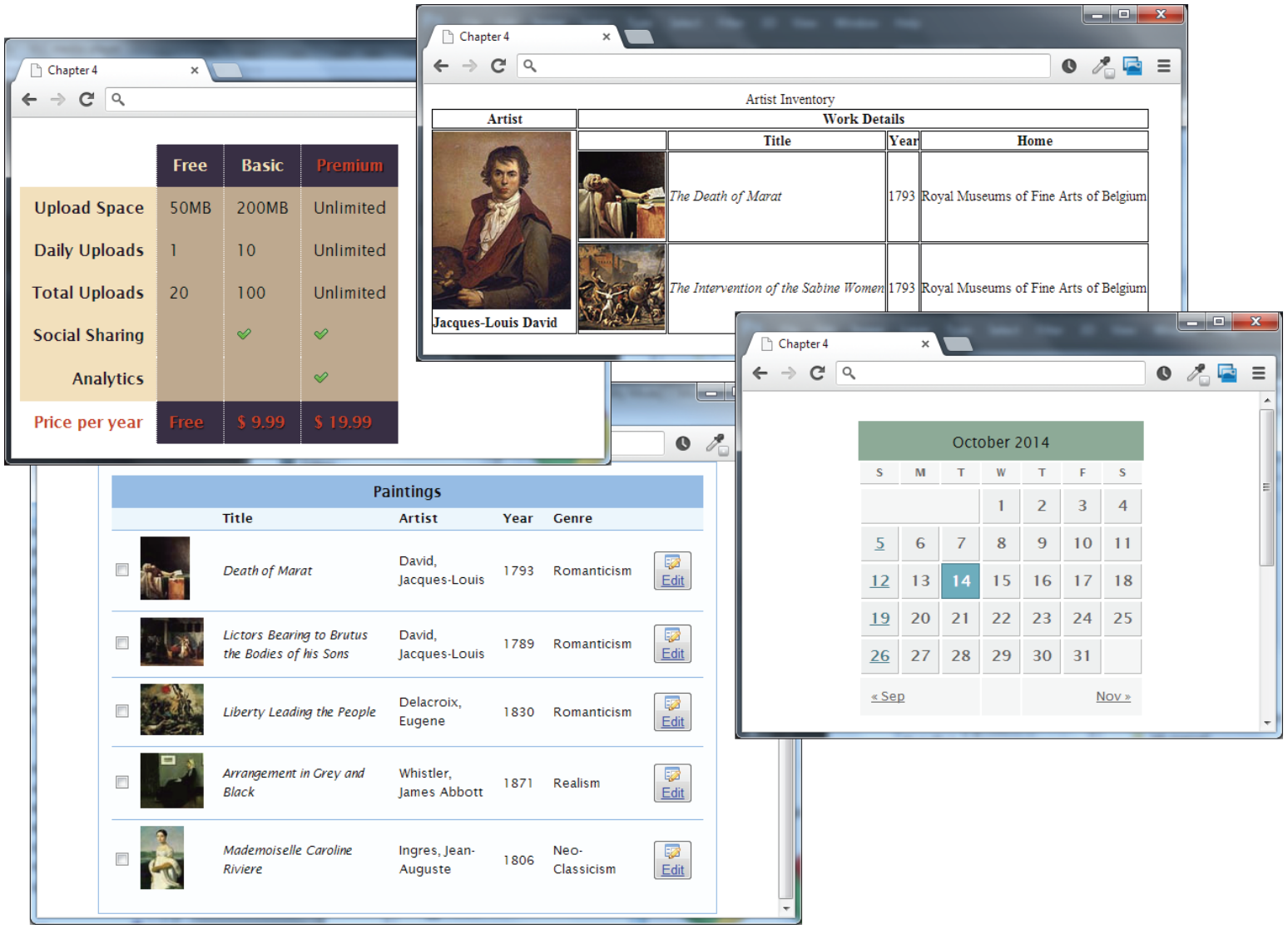
HTML Tables

HTML Tables

- 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

- 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
  ```

A basic Example

<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>

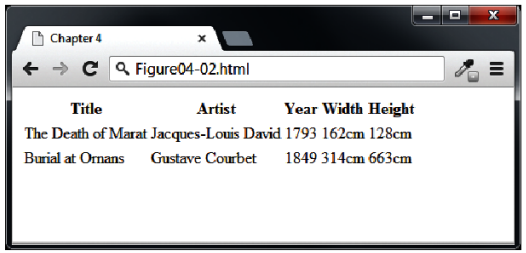
```
<table>
    <tr>
        <td>The Death of Marat</td>
        <td>Jacques-Louis David</td>
        <td>1793</td>
        <td>162cm</td>
        <td>128cm</td>
    </tr>
    <tr>
        <td>Burial at Ornans</td>
        <td>Gustave Courbet</td>
        <td>1849</td>
        <td>314cm</td>
       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

With Table Headings

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

th

```
<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>
```

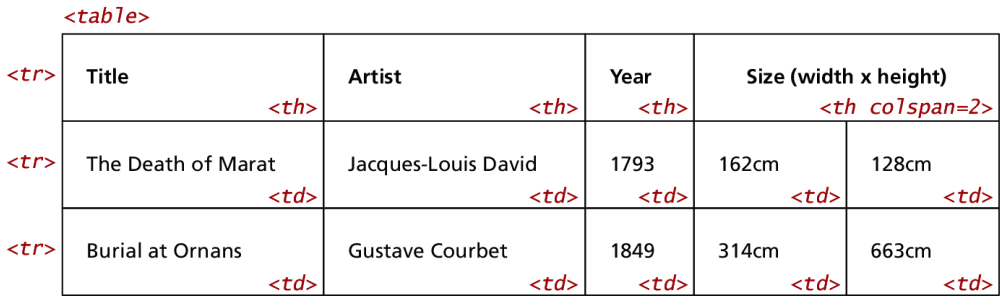


Why Table Headings

- A table heading <th>
  - Browsers tend to make the content within a <th> element bold
  - <th> element for accessibility (it helps those using screen readers)
  - Provides some semantic info about the row being a row of headers

Spanning Rows and Columns

- Each row must have the same number of <td> or <th> containers. If you want a given cell to cover several columns or rows,



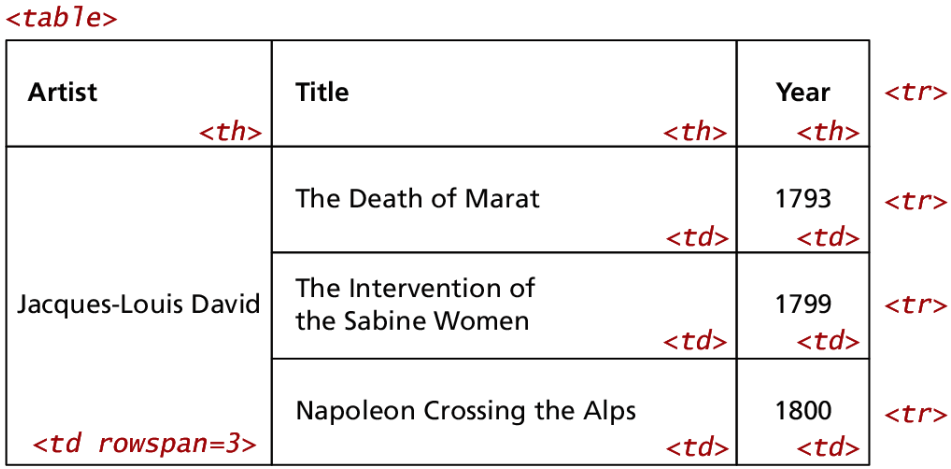
Notice that this row now only has four cell elements.

```
<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>
```

use the **colspan or rowspan attributes**

Using Tables for Layout

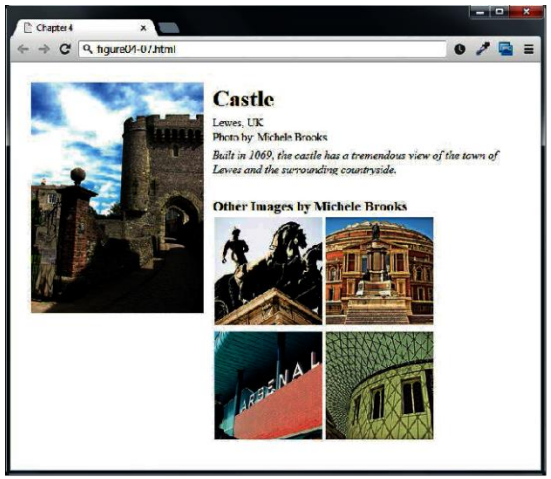
- It works in many situations
  - Popular in 1990s
  - Results in table bloat
  - Not semantic
  - Larger HTML pages
  - Browser quirks

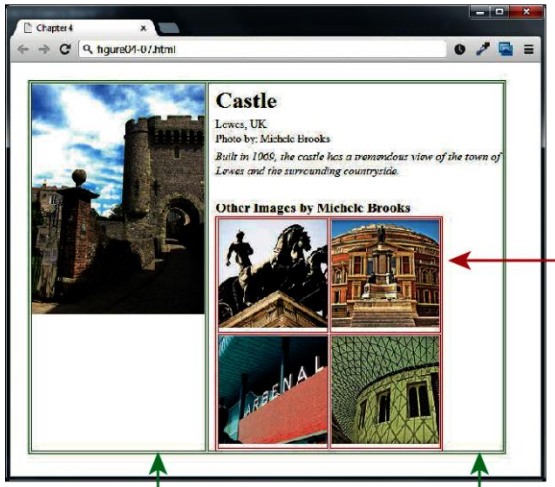



```
<table>
   <tr>
      <th>Artist</th>
      <th>Title</th>
      <th>Year</th>
   </tr>
   <tr>
      <td rowspan="3">Jacques-Louis David</td>
      <td>The Death of Marat</td>
      <td>1793</td>
   </tr>
   <tr>
      <td>The Intervention of the Sabine Women</td>
      <td>1799</td>
   </tr>
   <tr>
      <td>Napoleon Crossing the Alps</td>
      <td>1800</td>
   </tr>
   ...
</table>
```

Notice that these two rows now only have two cell elements.

Example Table layouts






```
<table>
  <tr>
    <td>
      <img src="images/959.jpg" alt="Castle"/>
    </td>
    <td>
      <h2>Castle</h2>
      <p>Lewes, UK</p>
      <p>Photo by: Michele Brooks</p>
      <p>Built in 1069, the castle has a tremendous
         view of the town of Lewes and the
         surrounding countryside.
      </p>

      <h3>Other Images by Michele Brooks</h3>

      <table>
        <tr>
          <td><img src="images/464.jpg" /></td>
          <td><img src="images/537.jpg" /></td>
        </tr>
        <tr>
          <td><img src="images/700.jpg" /></td>
          <td><img src="images/828.jpg" /></td>
        </tr>
      </table>
    </td>
  </tr>
</table>
```

Additional table tags

- <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.

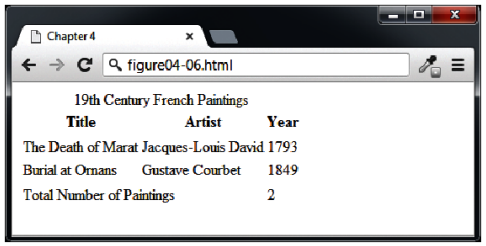
```
<table>
   <caption>19th Century French Paintings</caption>
   <col class="artistName" />
   <colgroup id="paintingColumns">
      <col />
      <col />
   </colgroup>

   <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>
```

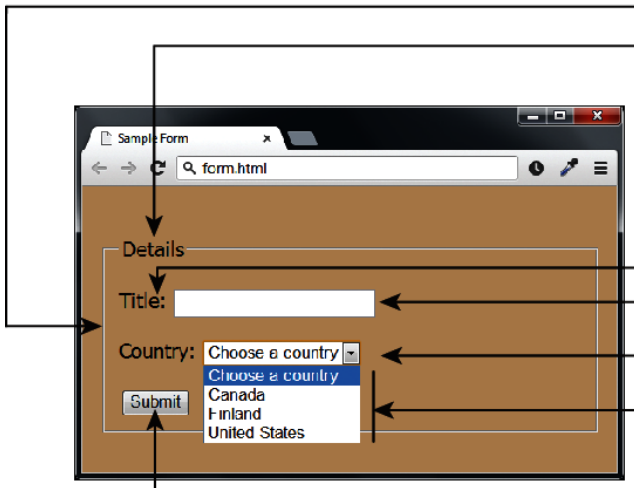


HTML Forms

CSLAB

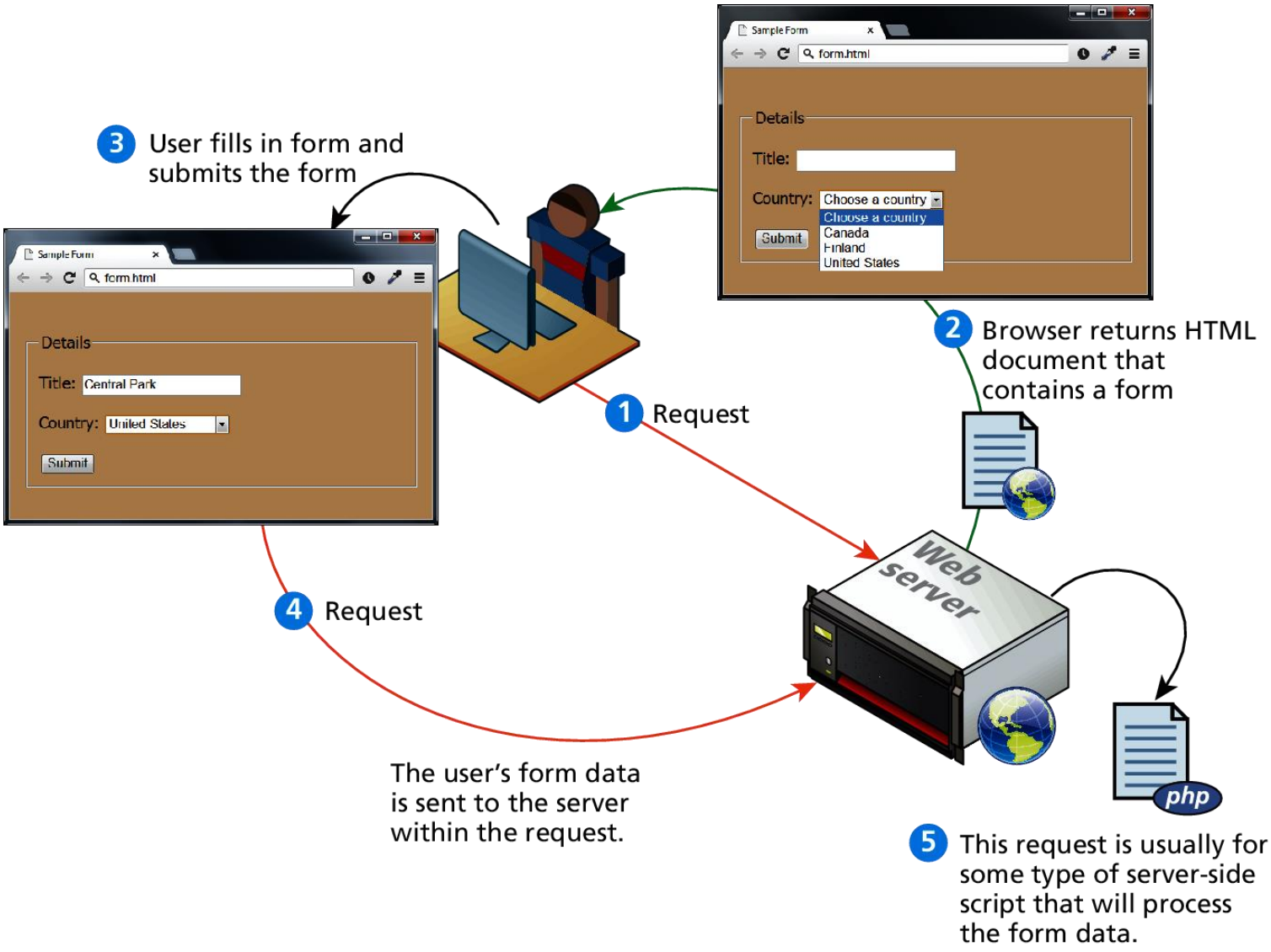
HTML Forms

- 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



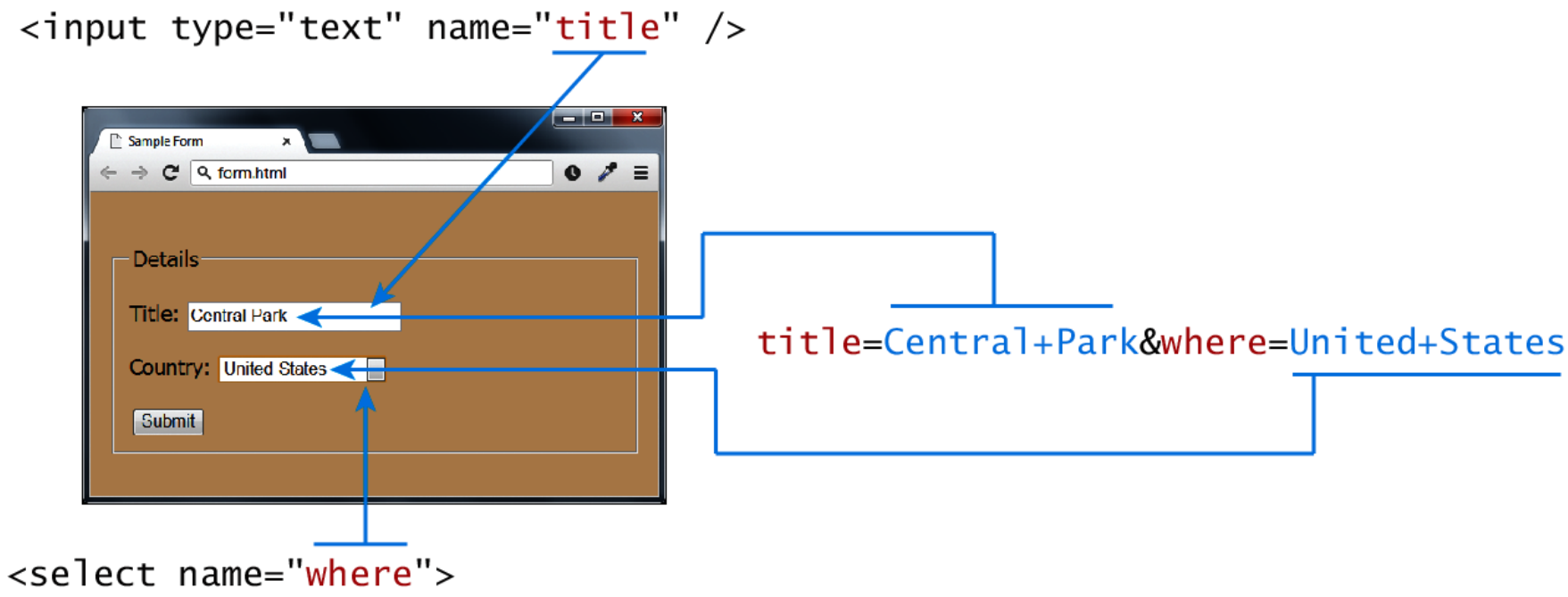

```
<form method="get" action="process.php">
 <fieldset>
  <legend>Details</legend>
  <p>
   <label>Title: </label>
   <input type="text" name="title" />
  </p>
  <p>
   <label>Country: </label>
   <select name="where">
    <option>Choose a country</option>
    <option>Canada</option>
    <option>Finland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

How forms interact with servers



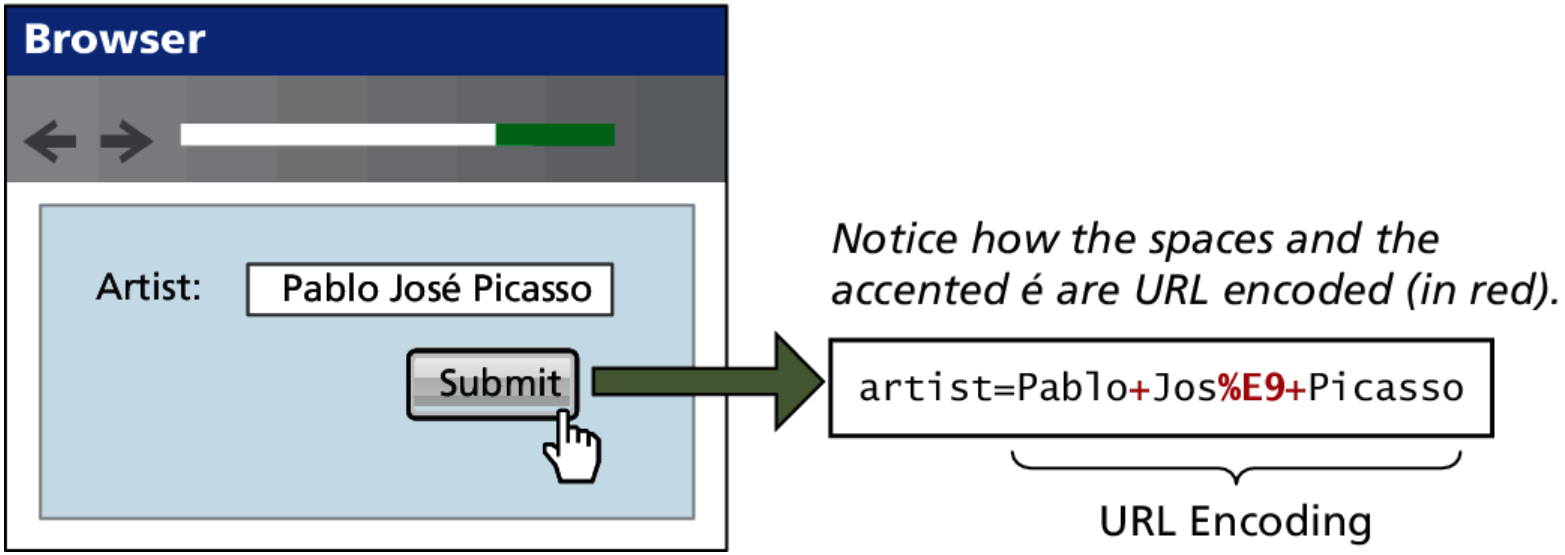
Query Strings

- At the end of the day, another string



URL encoding

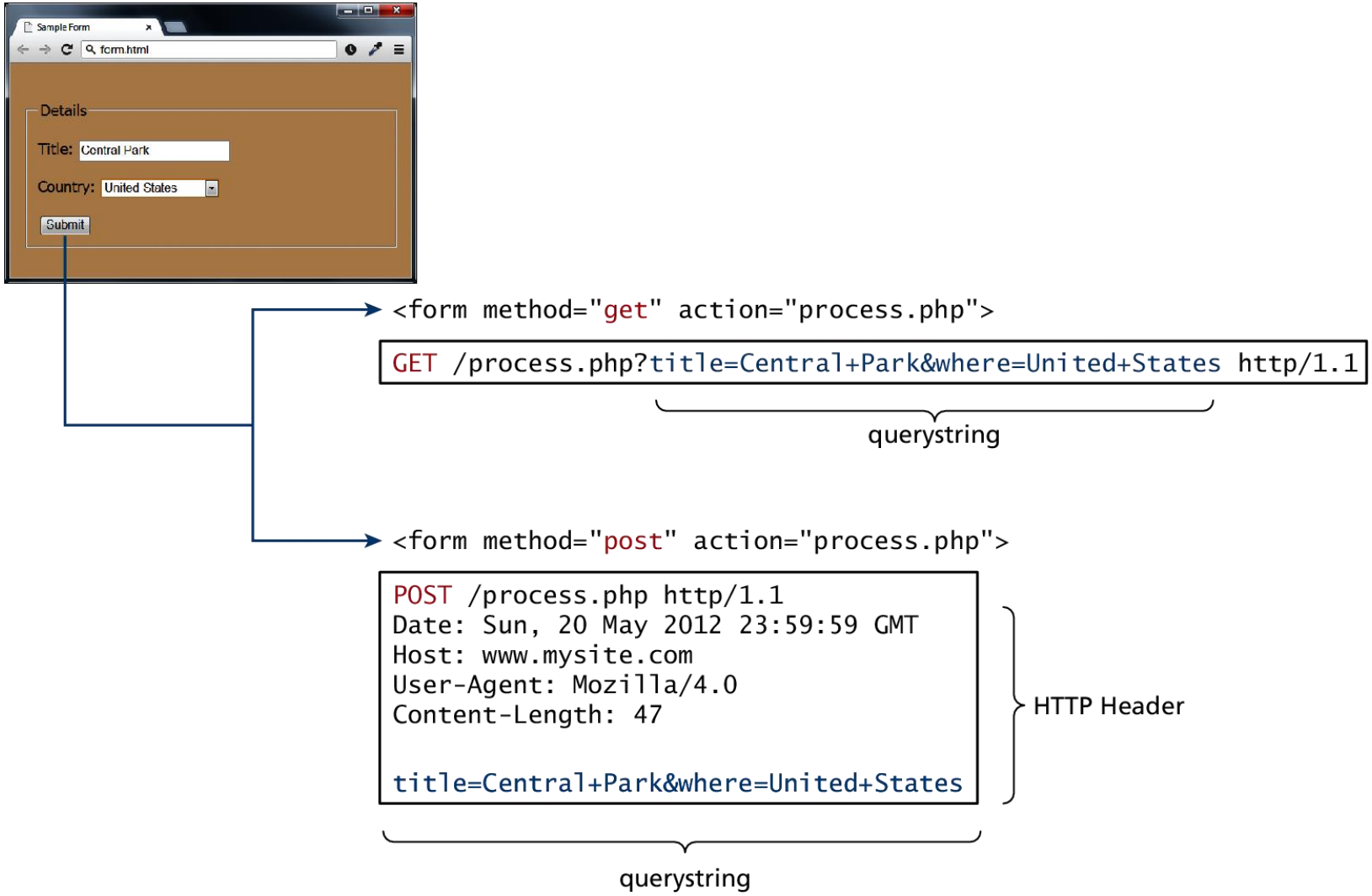
- Special symbols



<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


Sample Form
form.html
Details
Title: Central Park
Country: United States
Submit
<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.

Forms Control Elements

CSLAB

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

```
<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:

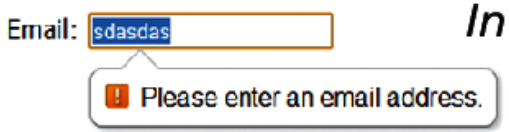
Password: ••••

Text Input Controls

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


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

In Opera



In Chrome

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


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

HTML5 advanced controls

Pattern attribute

```
<input type="text" ... placeholder="L#L #L#" pattern="[a-z][0-9][a-z] [0-9][a-z][0-9]"  />
```

datalist


```
<input type="text" name="city" list="cities"  />

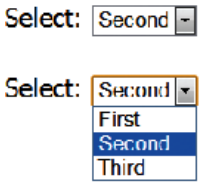
<datalist id="cities">
   <option>Calcutta</option>
   <option>Calgary</option>
   <option>London</option>
   <option>Los Angeles</option>
   <option>Paris</option>
   <option>Prague</option>
</datalist>
```

Select Lists

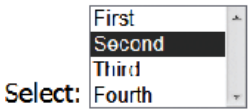
- **<select>** element is used to create a multiline box for selecting one or more items
  - The options are defined using the **<option>** element
  - can be hidden in a dropdown or multiple rows of the list can be visible
  - Option items can be grouped together via the **<optgroup>** element.

Select Lists

- Select List Examples




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

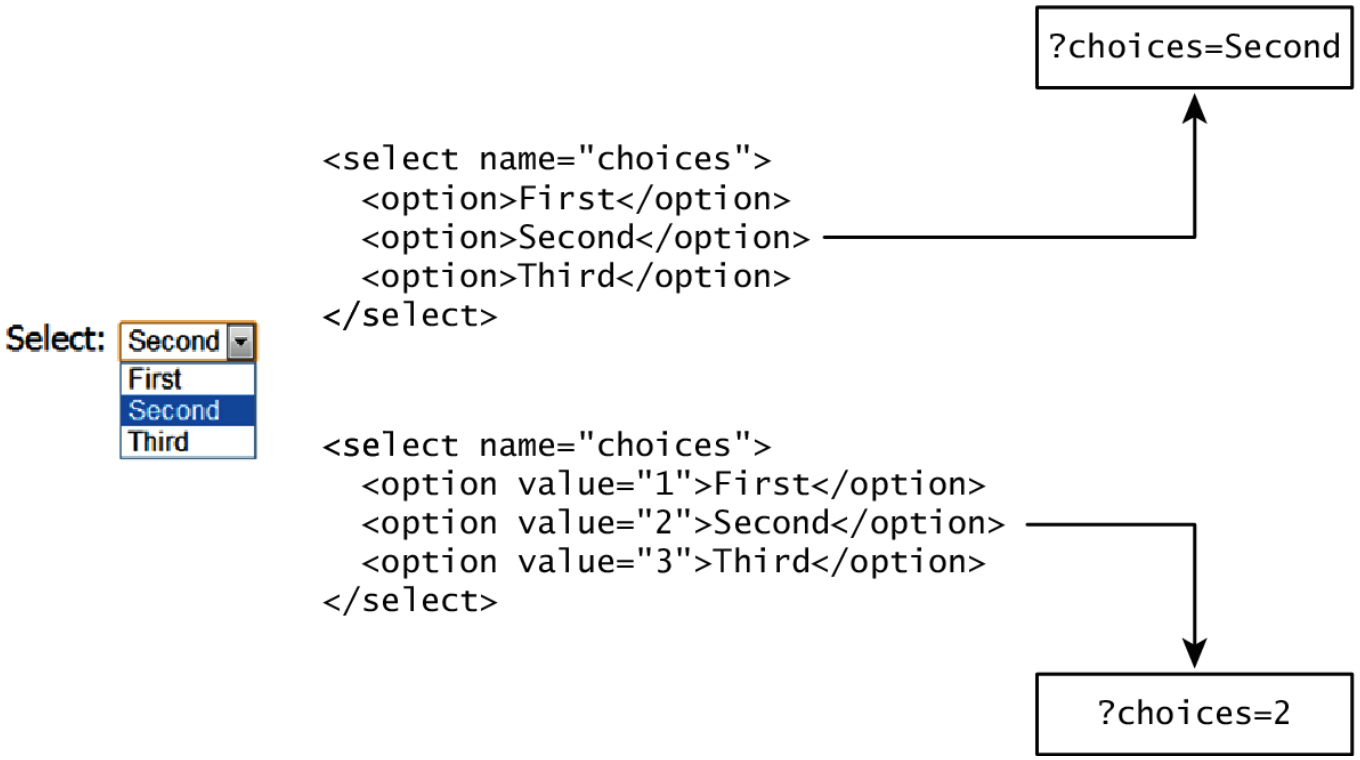



```
<select size="3" ... >
```


```
<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

- 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



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


```
<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 ☑

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

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

```
<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>
```

Number and Range

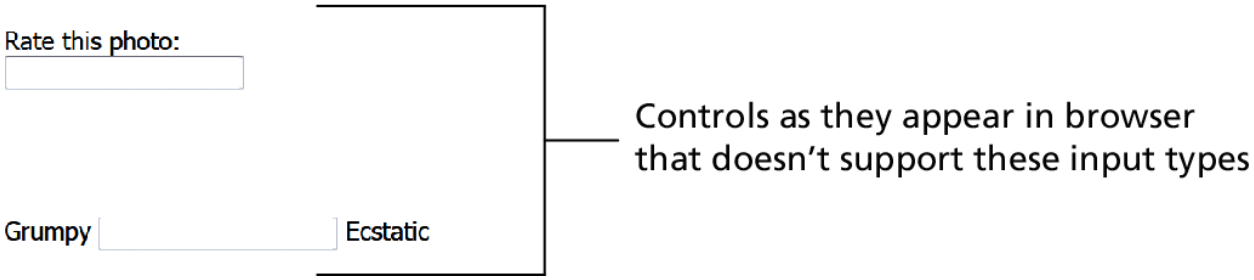
- Typically input values need be **validated**. Although server side validation is required, optional client side pre-validation is good practice.
- The number and range controls Added in HTML5 provide a way to input numeric values that **eliminates the need for JavaScript numeric validation!!!**

Rate this photo:
2

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

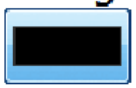
Grumpy Ecstatic

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

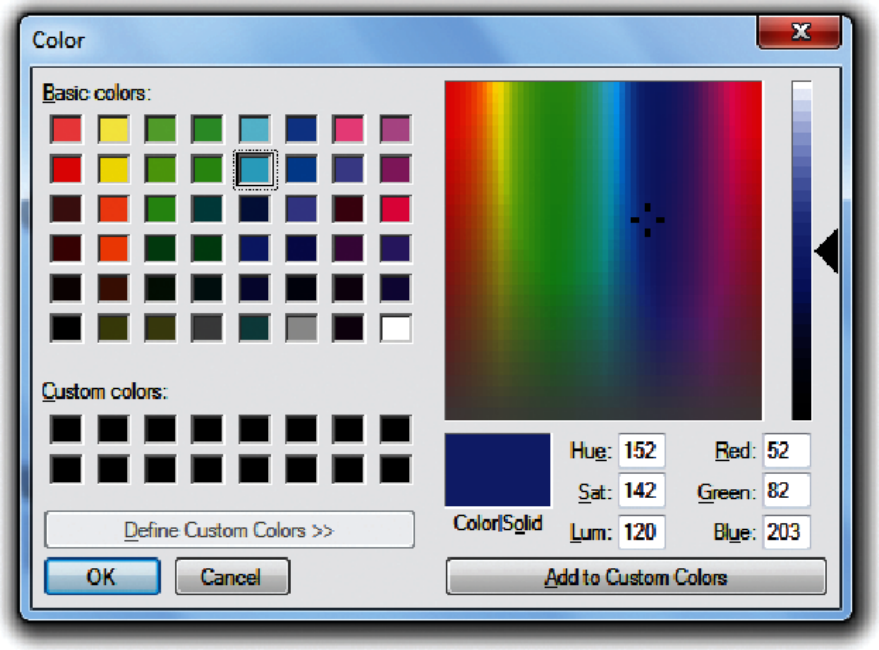


Color

Background Color:



```
<label>Background Color: <br/>
<input type="color" name="back" />
```



Background Color:

Control as it appears in browser that doesn't support this input type

Date and Time Controls

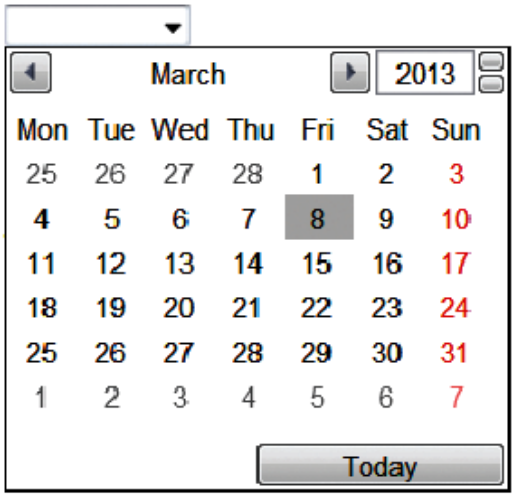
- Dates and times often need validation when gathering this information from a regular text input control.
- From a user's perspective, entering dates can be tricky as well: you probably have wondered at some point in time when entering a date into a web form, what format to enter it in, whether the day comes before the month, whether the month should be entered as an abbreviation or a number, and so on.

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##".

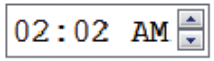
HTML5 Date and Time Controls

Date:




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

Time:




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

DateTime:

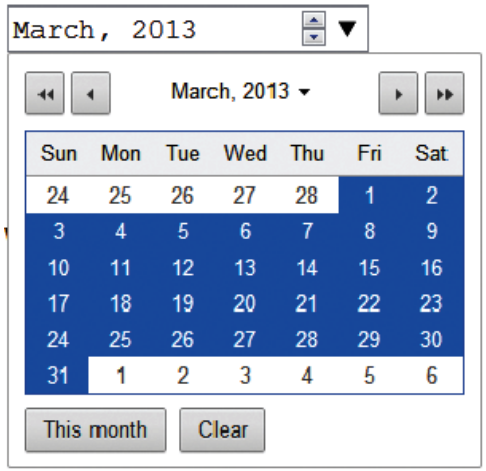

```
<input type="datetime" ... />
```

DateTime Local:


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

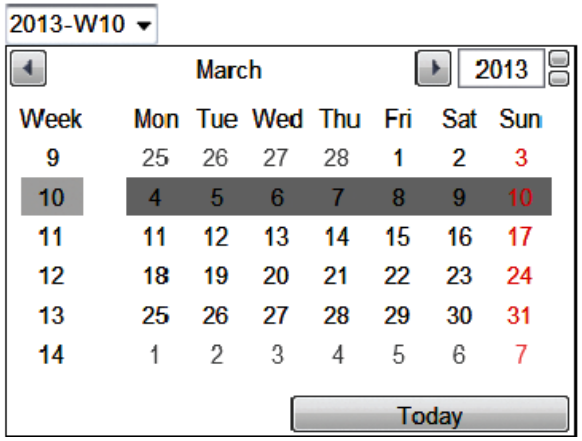
HTML5 Date and Time Controls

Month:




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

Week:




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