

HTML 1: Overview

Chapter 2

Brief History of HTML

Did we mention that this will be brief?

- Created by Tim Berners-Lee in 1991
- HTML's codification by the World-Wide Web Consortium (better known as the **W3C**) in 1997.
- And now **HTML 5 (2014)**

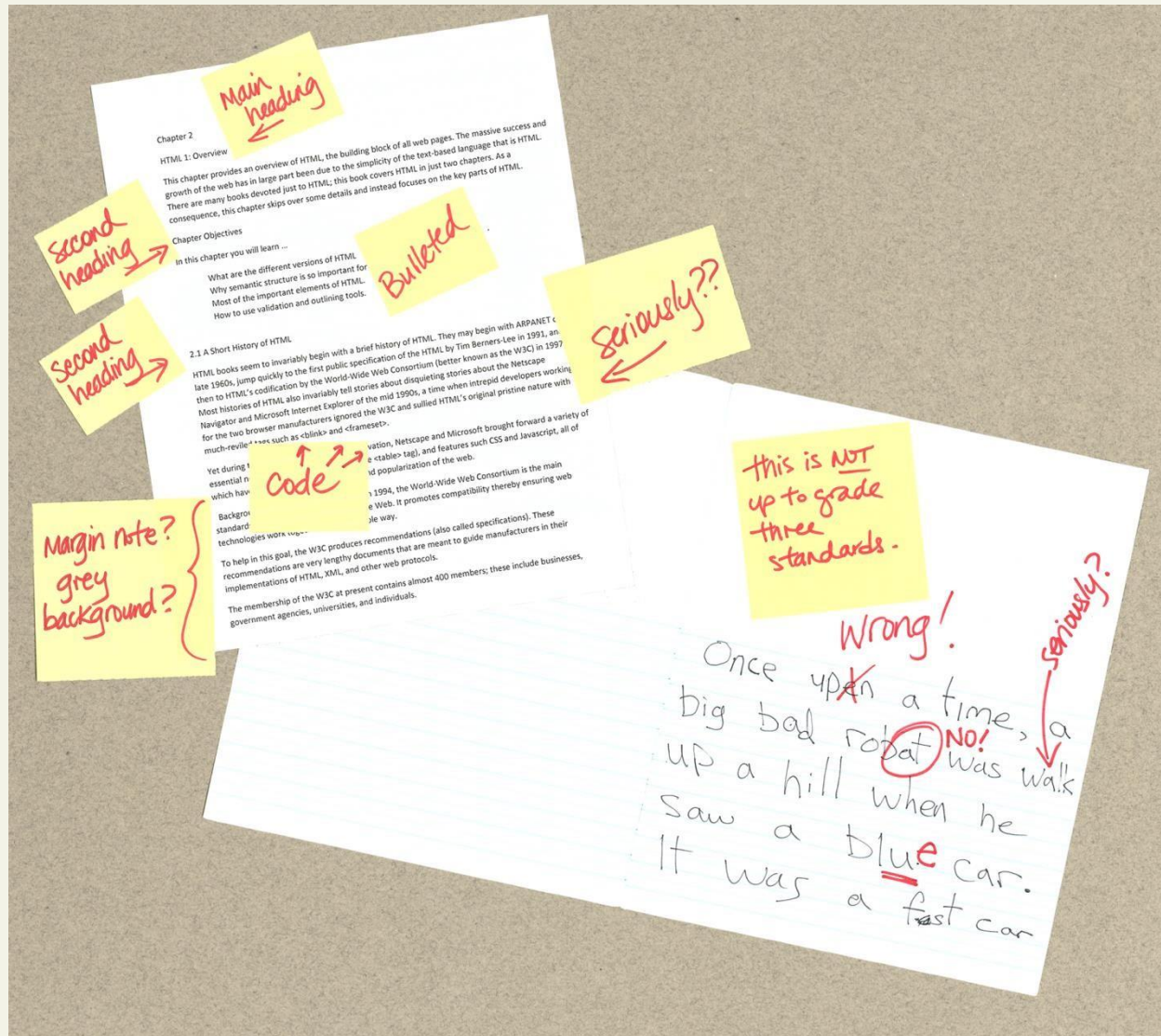
HTML Syntax

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.
- 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.

Sample ad hoc markup



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** (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.

What is the W3C?

Standards

- The W3C is the main standards organization for the World Wide Web.
- To promote 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.
- XHTML bridges the gap between HTML and XML.

XHTML

Partying like it's 1999

The goal of XHTML with its strict rules was to make page rendering more predictable by forcing web authors to create web pages without syntax errors.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type"
    content="text/html; charset=iso-8859-1" />
    <title>What is XHTML?</title>
  </head>
  <body>
    <p>The extensible Hypertext Markup Language (XHTML) is
    <p>So what is a markup language? A markup language cons
    <p>As the figure shows, XHTML is nothing like a web pag
  </body>
</html>
```



XHTML

You too can be strict

The XML-based syntax rules for XHTML are pretty easy to follow.

The main rules are:

- lowercase tag names,
- attributes always within quotes,
- and all elements must have a closing element (or be self-closing).

XHTML 2.0: a collapse?

Where did it go?

In the mid 2000s, XHTML 2.0 proposed a revolutionary and substantial change to HTML.

- backwards compatibility with HTML and XHTML 1.0 was dropped.
- Browsers would become significantly less forgiving of invalid markup.
- This lack of compatibility with XHTML 1.x and HTML 4 caused some early controversy in the web developer community. A ninth draft of XHTML 2.0 was expected to appear in 2009, but on July 2, 2009, the W3C decided to let the XHTML2 Working Group charter expire by that year's end, effectively halting any further development of the draft into a standard...

HTML5

The new hotness

By 2009, the W3C stopped work on XHTML 2.0 and instead adopted HTML5 (finalization in October 2014).



HTML5

Three main aims

There are three main aims to HTML5:

- Provide an **open, non-proprietary programming framework (via Javascript)** for creating rich web applications.
- Be backward compatible with the existing web.

HTML5

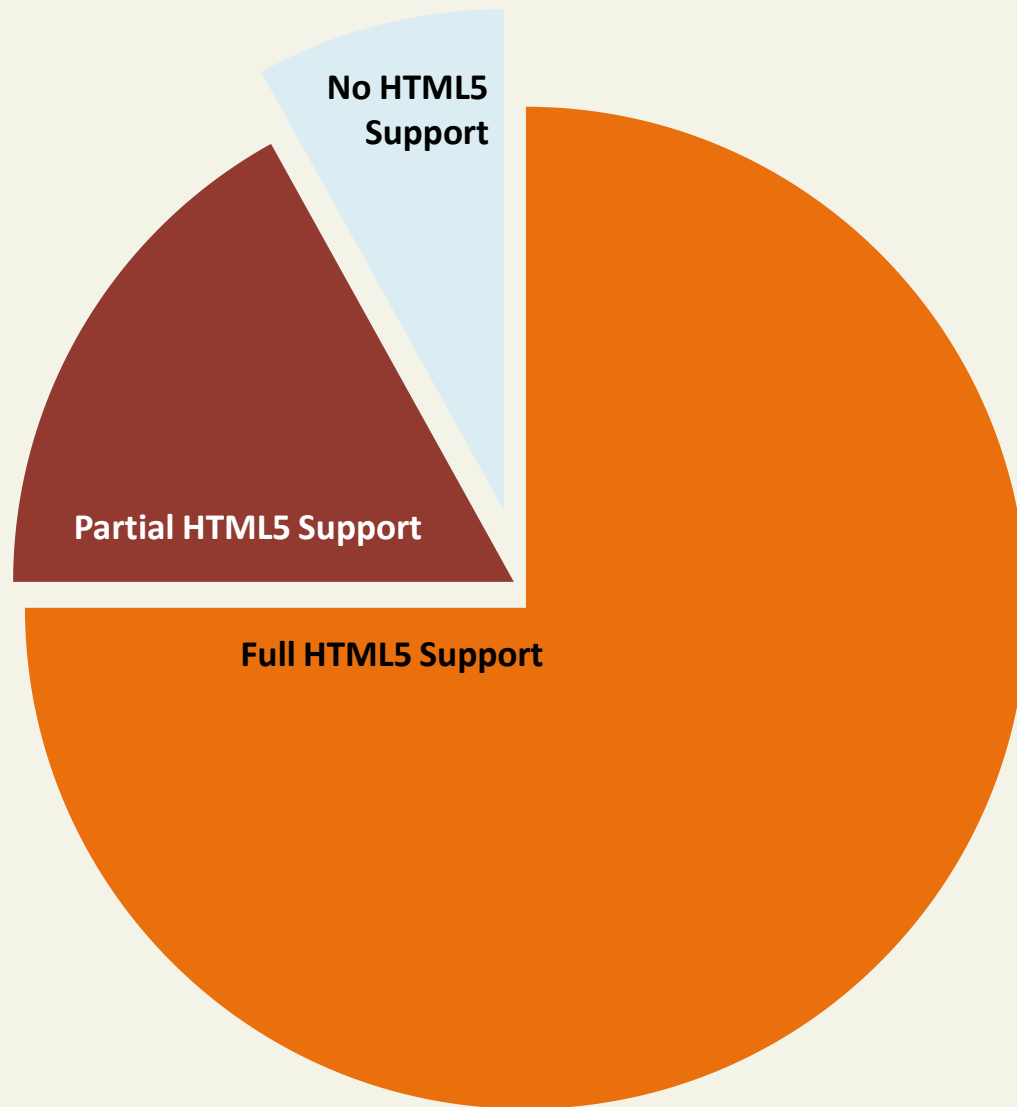
It evolves

All of the major browser manufacturers have at least partially embraced HTML5.

Certainly not all browsers and all versions support every feature of HTML5.

This is in fact by design. HTML in HTML5 is now a living language: that is, it is a language that evolves and develops over time.

HTML5 Support in Browsers



Section 2 of 6

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.

```

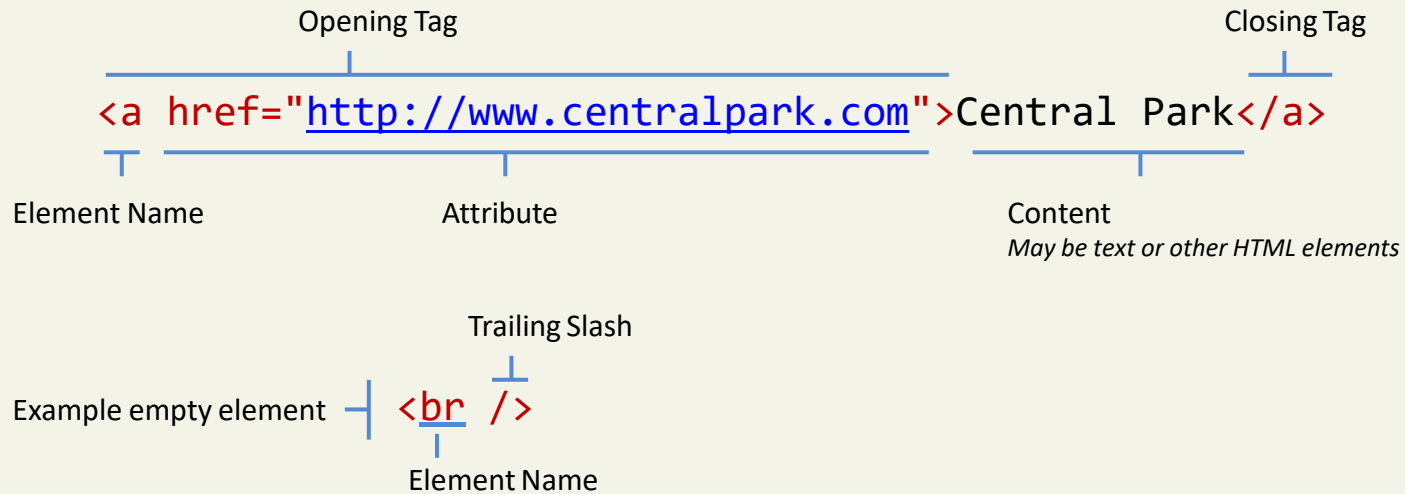
```

In XHTML, attribute values had to be enclosed in quotes; in HTML5, the quotes are optional.

What HTML lets you do

- Insert images using the `` tag
- Create links with the `<a>` tag
- Create lists with the ``, `` and `` tags
- Create headings with `<h1>`, `<h2>`, ..., `<h6>`
- And much more...

Elements and Attributes



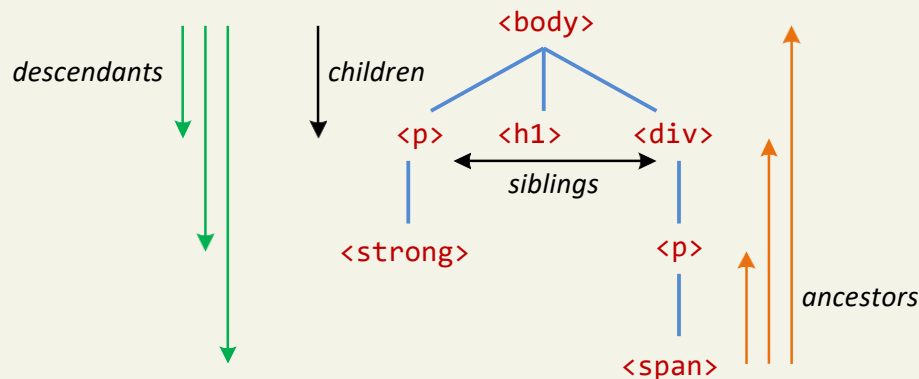
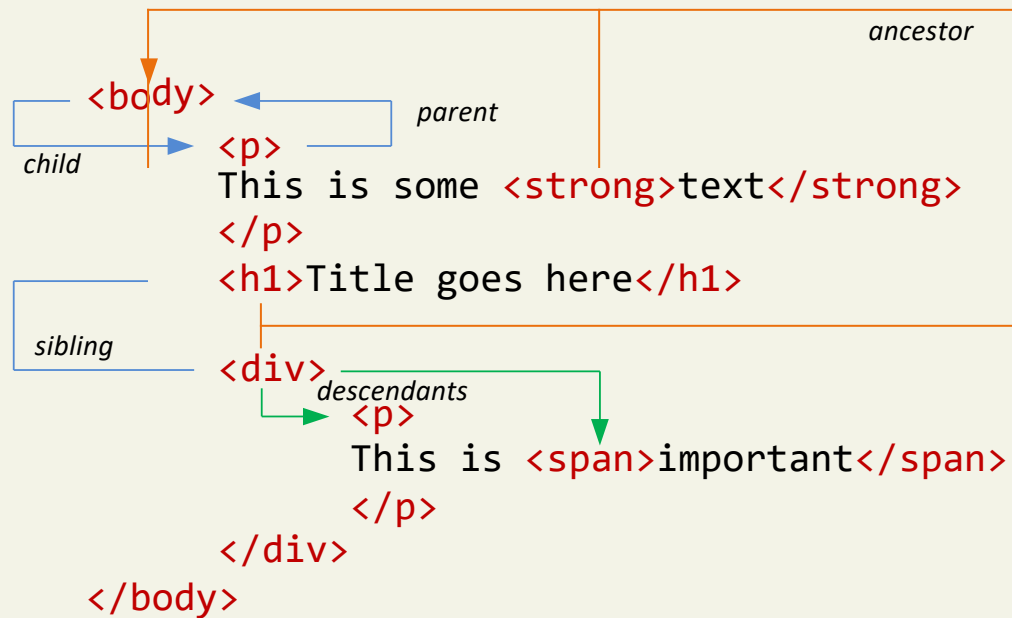
Nesting HTML elements

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

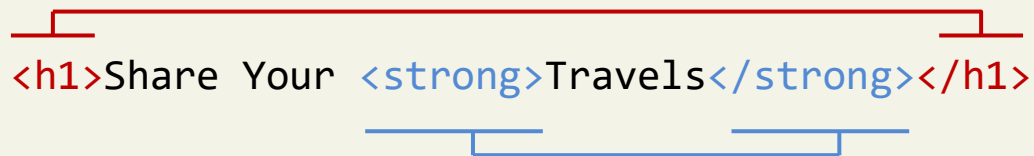


Nesting HTML elements

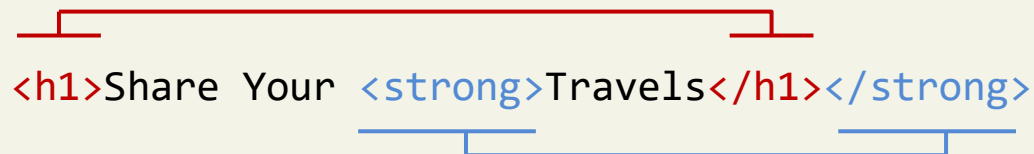
In order to properly construct a hierarchy of elements, your browser expects each HTML nested element to be properly nested.

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

Section 3 of 6

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 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.

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.

Section 4 of 6

STRUCTURE OF HTML

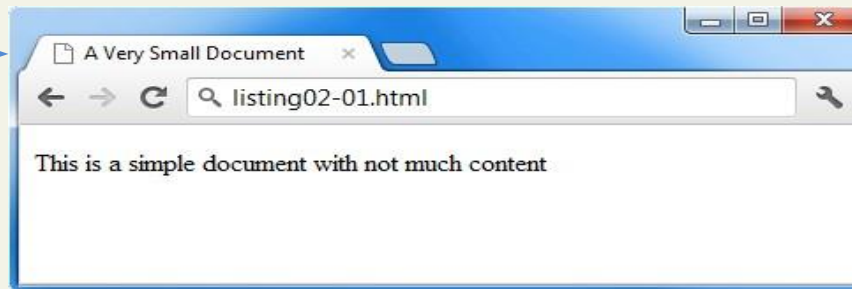
Simplest HTML document

1

```
<!DOCTYPE html>
```

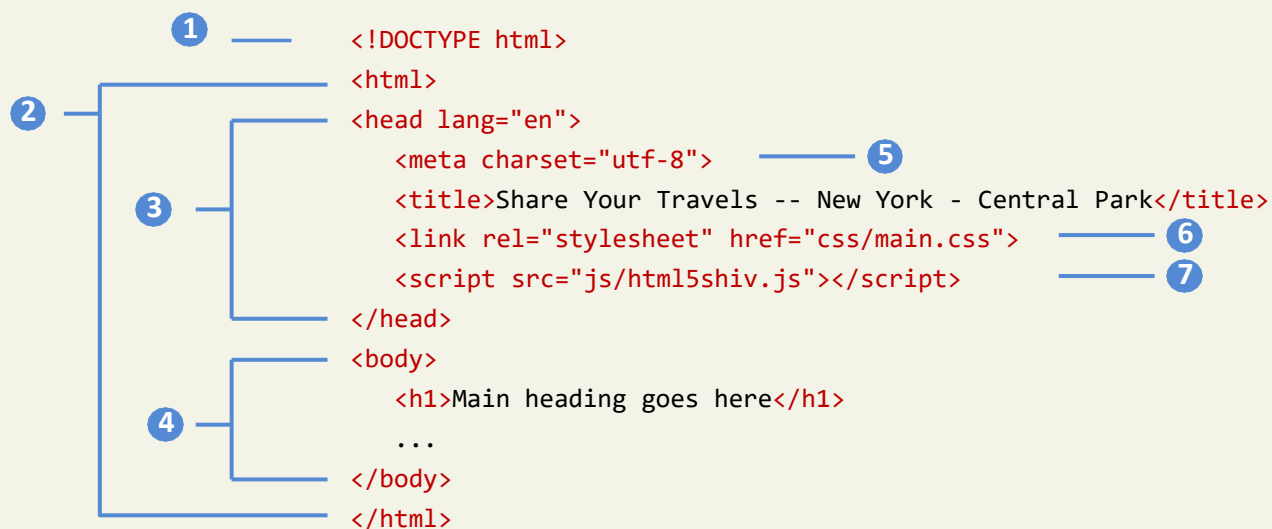
```
<title>A Very Small Document</title>
```

```
<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.

A more complete document



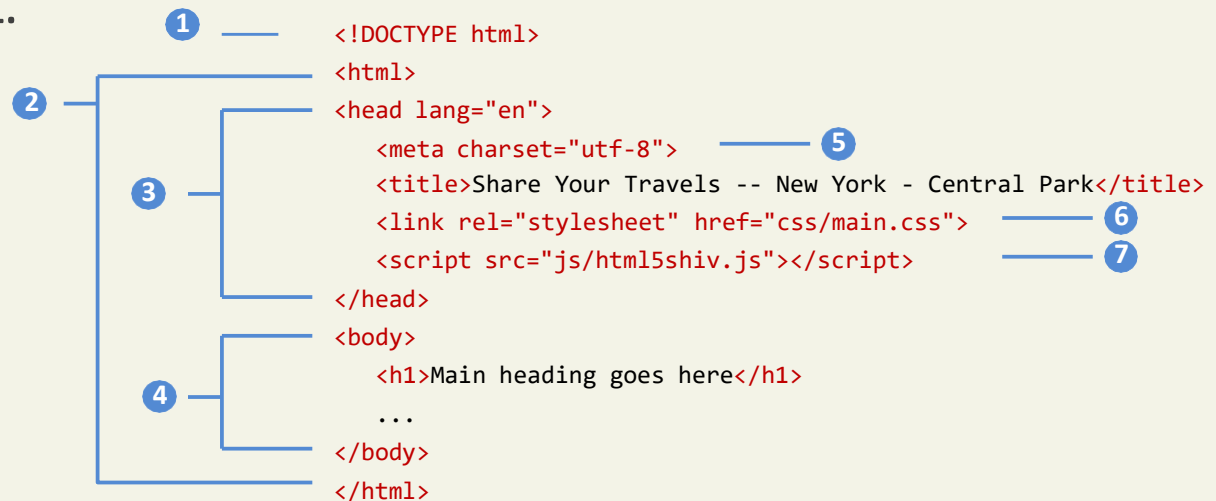
1

DOCTYPE

(short for **Document Type Definition**)

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.

- 2 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.

- 3 The head contains descriptive elements *about* the document
- 4 The body contains content that will be displayed by the browser.

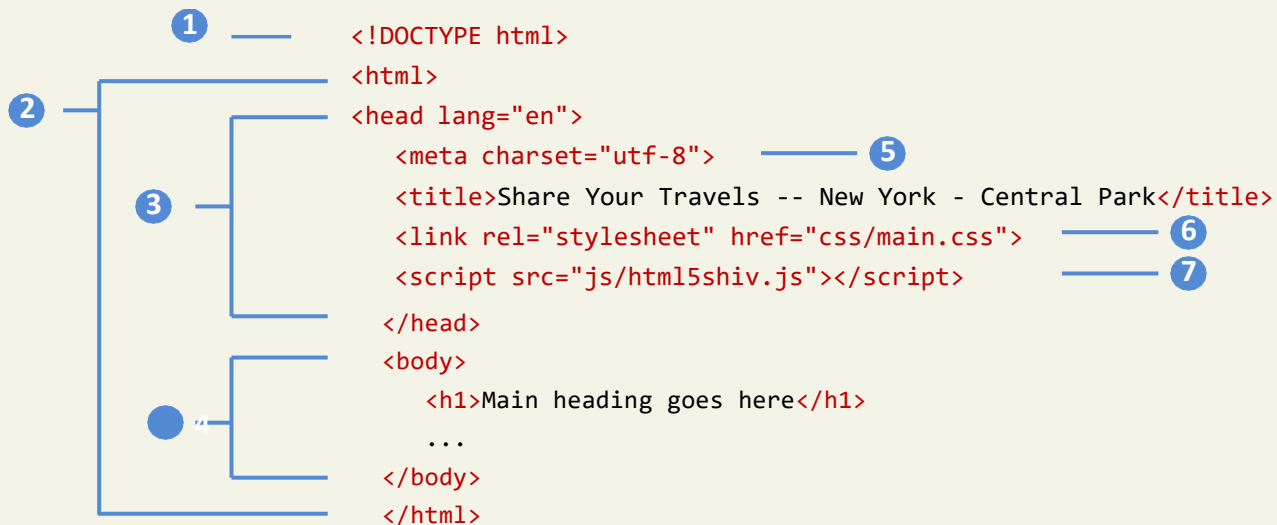


Inside the head

There are no brains

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

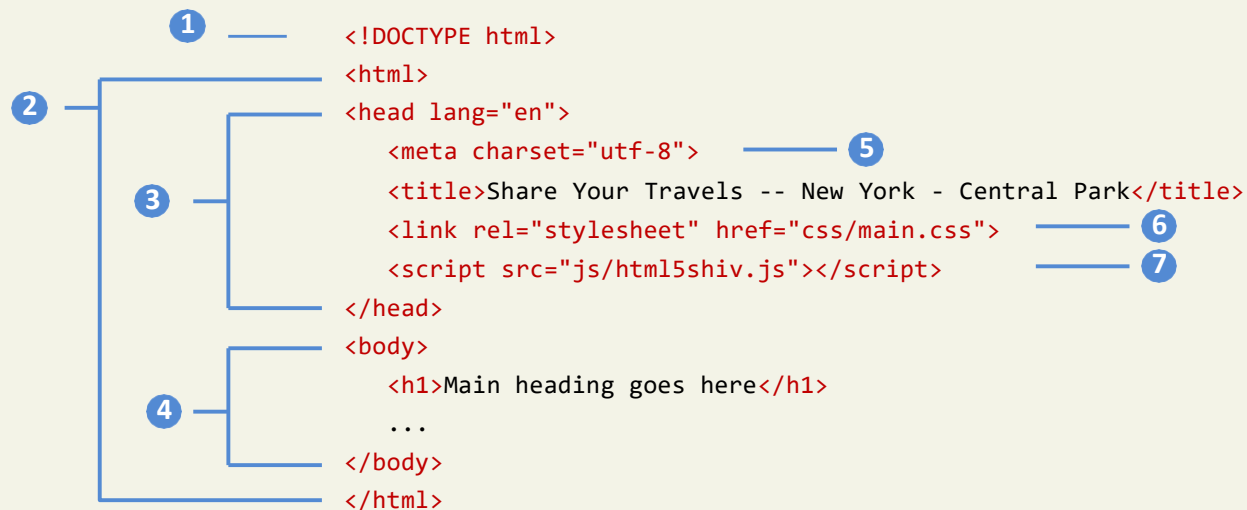
- 5 The first of these is the `<meta>` element. Our example declares that the character encoding for the document is UTF-8.



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.



Section 5 of 6

QUICK TOUR OF HTML

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`

`Central Park`

`New York City was taken on October 22, 2011 with a`

`Canon EOS 30D camera.`

`</p>`

``

`<h3>Reviews</h3>`

`<div>`

`<p>By Ricardo on <time>September 15, 2012</time></p>`

`<p>Easy on the HDR buddy.</p>`

`</div>`

`<div>`

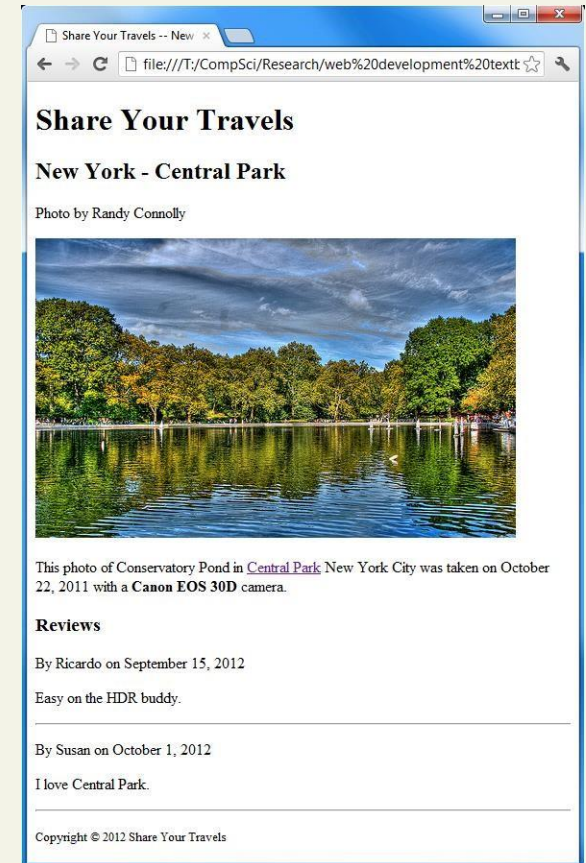
`<p>By Susan on <time>October 1, 2012</time></p>`

`<p>I love Central Park.</p>`

`</div>`

`<p><small>Copyright © 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 more importance.

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

My Term Paper Outline

1. Introduction

2. Background

2.1 Previous Research

2.2 Unresolved issues

3. My Solution

3.1 Methodology

3.2 Results

3.3 Discussion

4. Conclusion

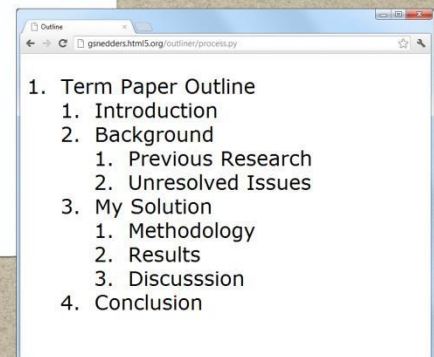
```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="utf-8">
  <title>Term Paper Outline</title>
</head>
<body>
  <h1>Term Paper Outline</h1>

  <h2>Introduction</h2>

  <h2>Background</h2>
  <h3>Previous Research</h3>
  <h3>Unresolved Issues</h3>

  <h2>My Solution</h2>
  <h3>Methodology</h3>
  <h3>Results</h3>
  <h3>Discussion</h3>

  <h2>Conclusion</h2>
</body>
</html>
```



2 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.

`
`, ``, `<a>`

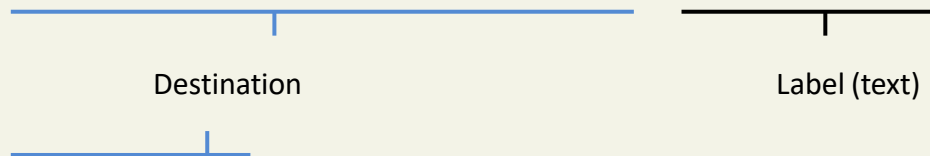
3 Links

`<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>
```



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



Types of Links

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

- Links to external sites (or to individual resources such as images or movies on an external site).
- Links to other pages or resources within the current site.
- Links to other places within the current page.
- Links to particular locations on another page.
- Links that are instructions to the browser to start the user's email program.
- Links that are instructions to the browser to execute a Javascript function.

Different link destinations

Link to external site

`Central Park`

Link to resource on external site

`Central Park`

Link to another page on same site as this page

`Home`

Link to another place on the same page

`Go to Top of Document`

Link to specific place on another page

`Reviews for product X`

Link to email

`Someone`

Link to javascript function

`See This`

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

`Call toll free (800) 922-0579`

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.
 - `http://website/chap1/img1.png`

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**.

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.

URL Relative Referencing

If all the resources for the site reside within the same **directory** (also referred to as a **folder**), then you can reference those other resources simply via their filename.

However, most real-world sites contain too many files to put them all within a single directory.

For these situations, a relative pathname is required along with the filename.

The **pathname** tells the browser where to locate the file on the server.

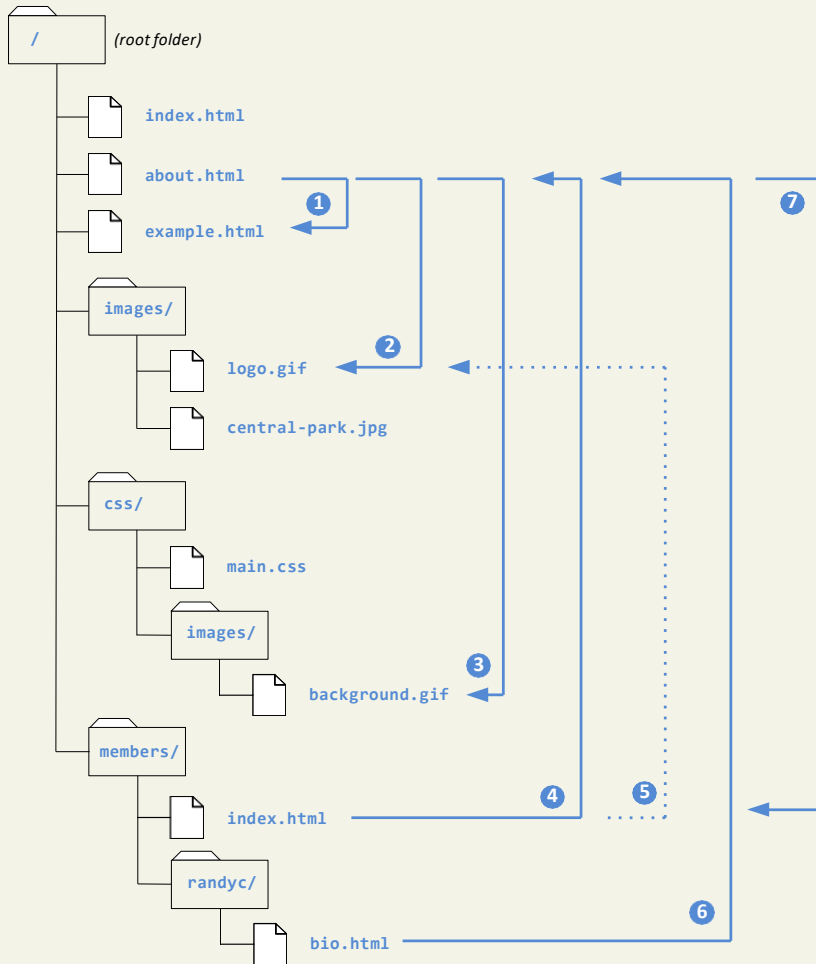
Pathnames

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

Share-Your-Travels



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: <code></code>
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: <code></code>
3 Grandchild/Descendant Directory To link to a file that is multiple subdirectories <i>below</i> 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: <code></code>
4 Parent/Ancessor Directory Use <code>"../"</code> to reference a folder <i>above</i> the current one. If trying to reference a file several levels above the current one, simply string together multiple <code>"../"</code> .	To link to about.html from index.html in members , use: <code></code> To link to about.html from bio.html , use: <code></code>

Images

While the `` tag is the oldest method for displaying an image.

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.

```

```

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.

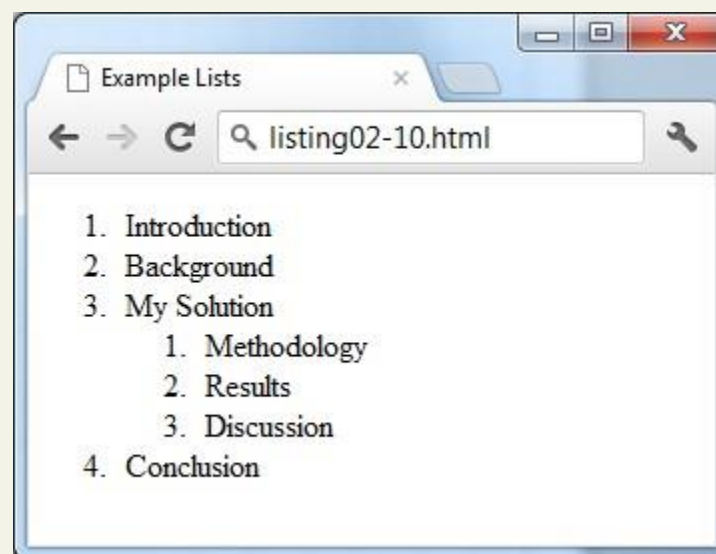
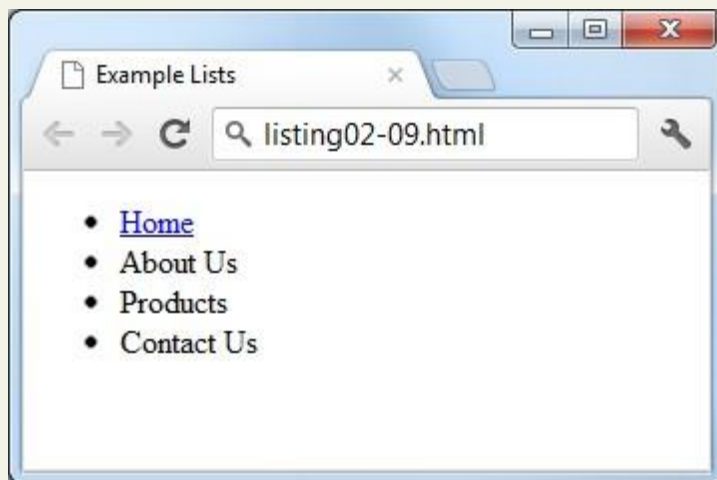
Definition lists. Collection of name and definition pairs. These tend to be used infrequently. Perhaps the most common example would be a FAQ list.

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 ©

Section 6 of 6

HTML 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 `<div>` elements.

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

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

XHTML versus HTML5

```
<div id="header">
  <div id="top-navigation">
    ...
  </div>
</div>
```

```
<div id="main">
  <div id="left-navigation">
    ...
  </div>
  <div class="content">
    <div class="story">
      ...
    </div>
  </div>
</div>
```

```
<div class="story-photo">
  <img ... class="blog-photo" />
  <p class="photo-caption">...</p>
</div>
```



```
<header>
  <nav>
    ...
  </nav>
</header>
```

```
<div id="main">
  <nav>
    ...
  </nav>
  <article>
    <section>
      ...
    </section>
  </article>
</div>
```

```
<figure>
  <img ... class="blog-photo" />
  <figcaption>...</figcaption>
</figure>
```

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

Header and Footer

`<header>` `<footer>`

The typical footer contains less important material, such as

- smaller text versions
- 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>
  
  <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>
```

HTML Tables and Forms

Chapter 4

Section 1 of 6

INTRODUCING TABLES

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

The collage shows four browser windows, each displaying a different table structure:

- Chapter 4 - Pricing Table:** A table with 4 columns: Feature, Free, Basic, Premium. It lists features like Upload Space, Daily Uploads, Total Uploads, Social Sharing, Analytics, and Price per year.
- Chapter 4 - Artist Inventory:** A table with 4 columns: Artist, Title, Year, Home. It lists works by Jacques-Louis David, including 'The Death of Marat' and 'The Intervention of the Sabine Women'.
- Chapter 4 - Paintings:** A table with 5 columns: Title, Artist, Year, Genre, and an Edit button. It lists various paintings like 'Death of Marat', 'Lictors Bearing to Brutus the Bodies of his Sons', 'Liberty Leading the People', 'Arrangement in Grey and Black', and 'Mademoiselle Caroline Riviere'.
- Chapter 4 - Calendar:** A calendar for October 2014, showing days of the week and dates.

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>
```

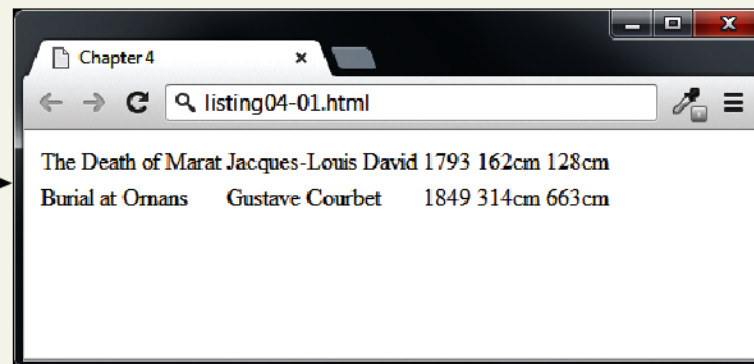


content

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>
```

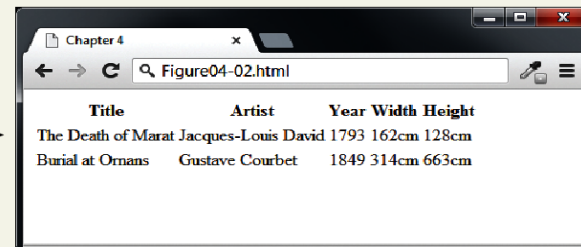


With Table Headings

<table>	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>

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

Span Span Span a Row

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

Title	Artist	Year	Size (width x height)	
The Death of Marat	Jacques-Louis David	1793	162cm	128cm
Burial at Ornans	Gustave Courbet	1849	314cm	663cm

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

Section 3 of 6

INTRODUCING FORMS

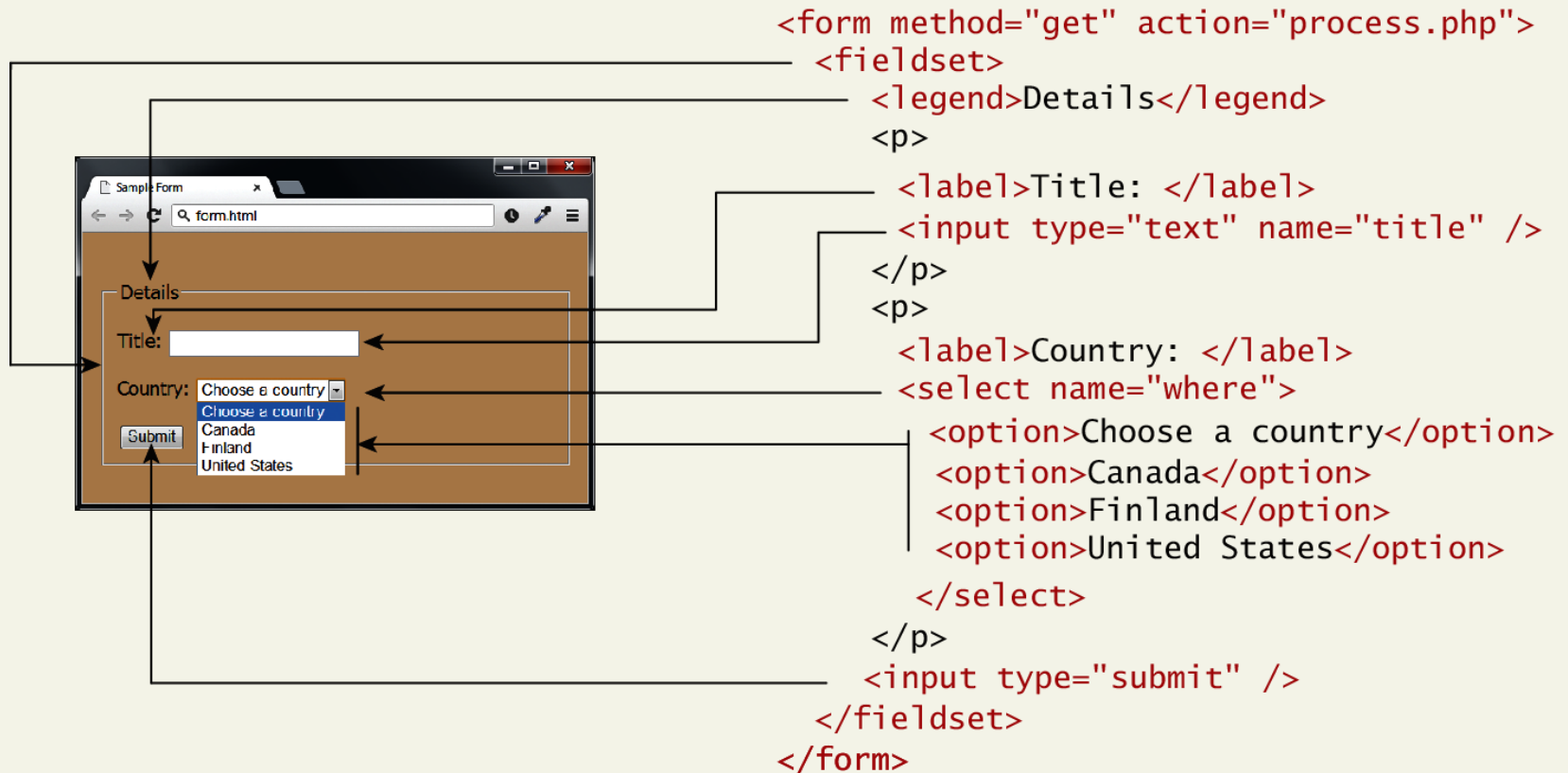
HTML Forms

Richer way to interact with server

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

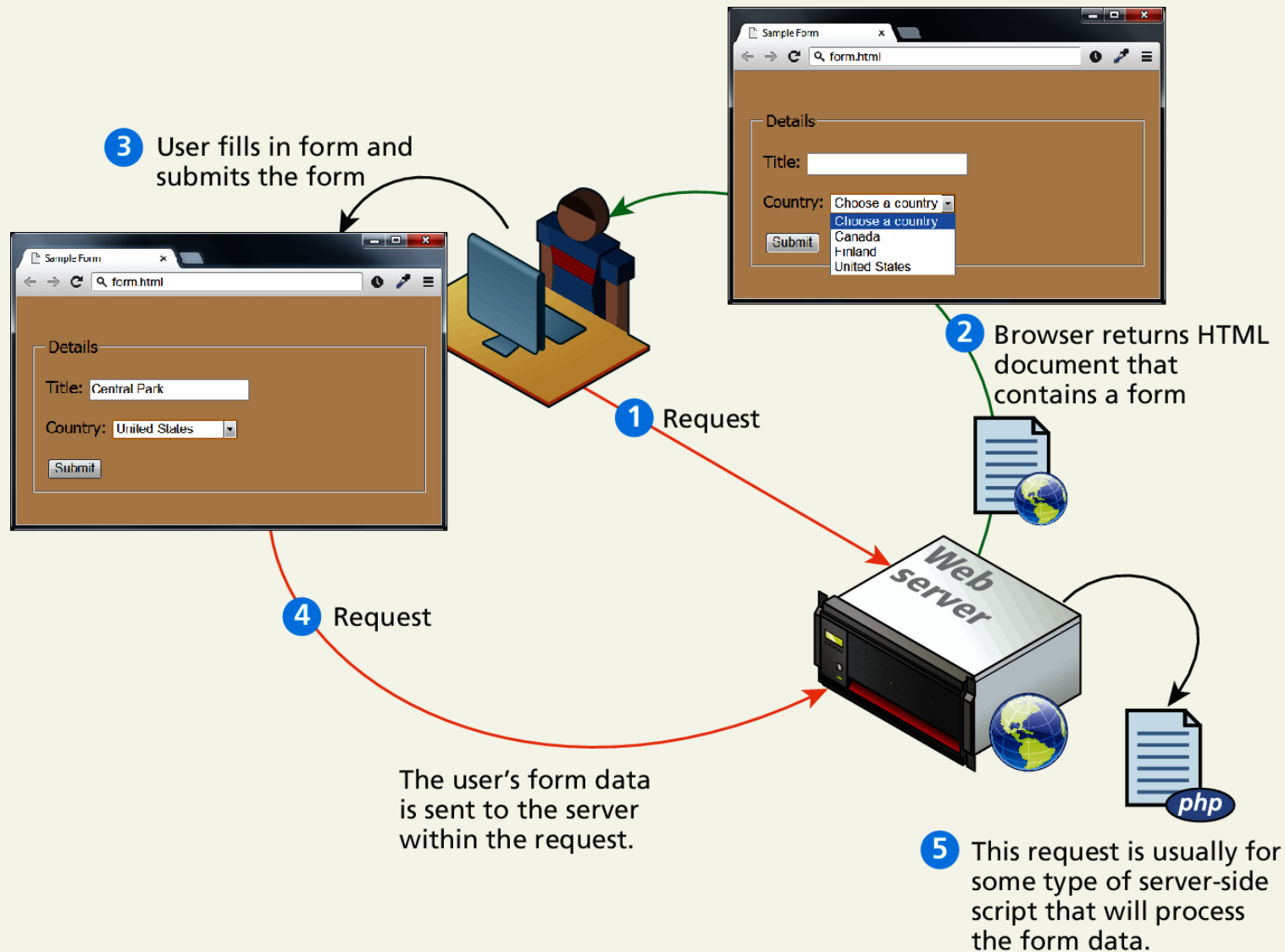
- Forms provide rich mechanisms like:
 - Text input
 - Password input
 - Options Lists
 - Radio and check boxes

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>
```

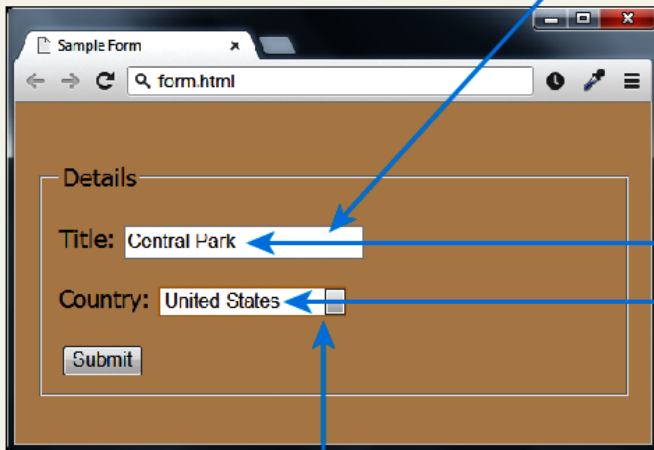
How forms interact with servers



Query Strings

At the end of the day, another string

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



A screenshot of a web browser window titled 'Sample Form'. The browser's address bar shows 'form.html'. The form itself has a brown background and contains two input fields: 'Title' with the value 'Central Park' and 'Country' with the value 'United States'. A 'Submit' button is located below the 'Country' field. Blue arrows point from the 'title' attribute in the code above to the 'Title' input field, and from the 'where' attribute in the code below to the 'Country' input field. A blue line also connects the 'title' attribute to the 'title' part of the query string on the right.

title=Central+Park&where=United+States

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

URL encoding

Special symbols

Browser

← →

Artist:

Notice how the spaces and the accented é are URL encoded (in red).

artist=Pablo+Jos%E9+Picasso

URL Encoding

<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

A screenshot of a web browser window titled 'Sample Form'. The address bar shows 'form.html'. The form contains a 'Title' text input with 'Central Park' and a 'Country' dropdown menu with 'United States' selected. A 'Submit' button is at the bottom. A blue line connects the 'Submit' button to the GET request example below.

`<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

- 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, in body section of request.
- Submitted data is not stored in cache, history, or bookmarks.

Section 4 of 6

FORMS CONTROL ELEMENTS

Form-Related HTML Elements

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

Text Input Controls

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

Text Input Controls

Classic

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

Text:

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

TextArea:

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

TextArea:

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

Password: Password:

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]" />
```

Postal:

Postal:

! Please match the requested format.

datalist

Search City:

Paris
Prague

```
<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

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



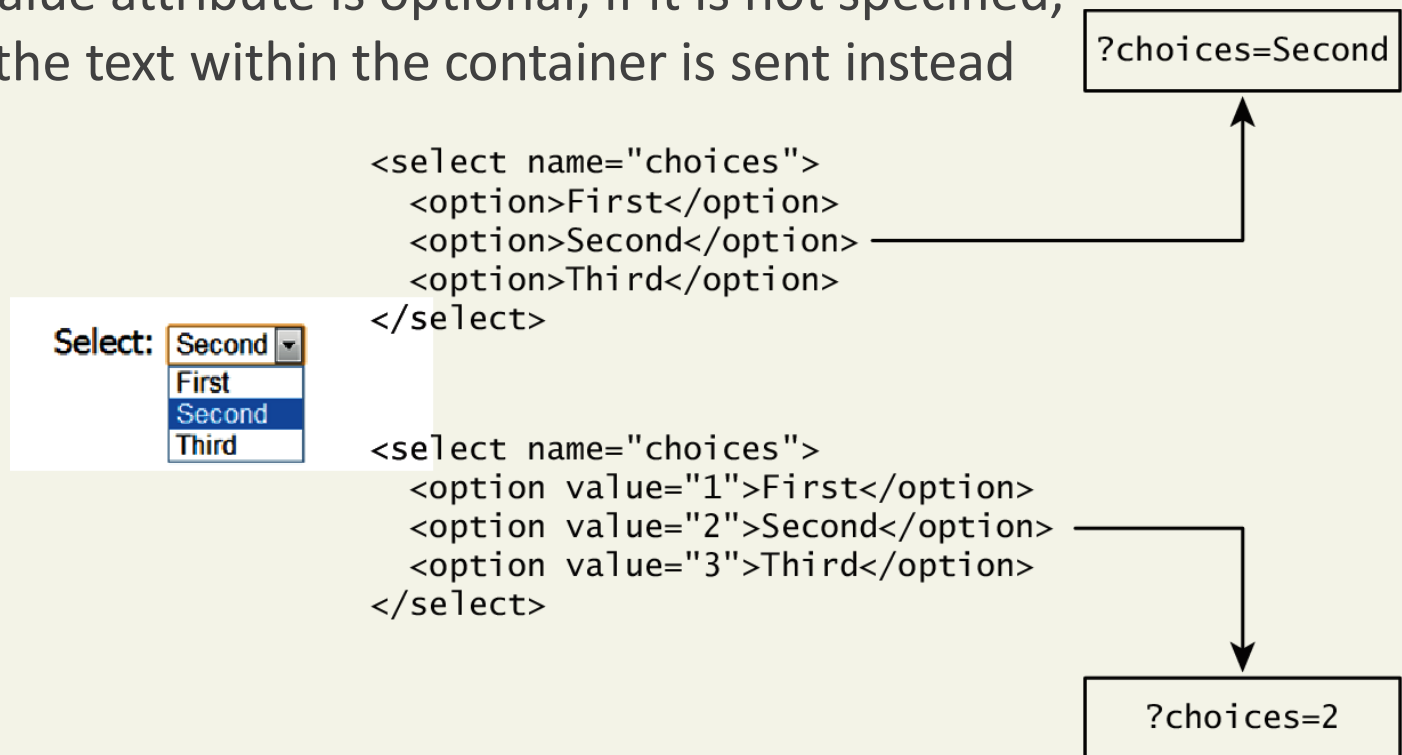
```
<select ... >
  <optgroup label="North America">
    <option>Calgary</option>
    <option>Los Angeles</option>
  </optgroup>
  <optgroup label="Europe">
    <option>London</option>
    <option>Paris</option>
    <option>Prague</option>
  </optgroup>
</select>
```

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



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 ☒

```
<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

