



HTML Part Two

COSC 484 Web-based Programming Spring 2006

David S. Gregory
dgregory@towson.edu

Copyright © 2003-2006 David S. Gregory, All rights reserved.

Agenda

- Adding images to documents
- Frames
- Forms

IMG: Embedding Images

- Example

```

```

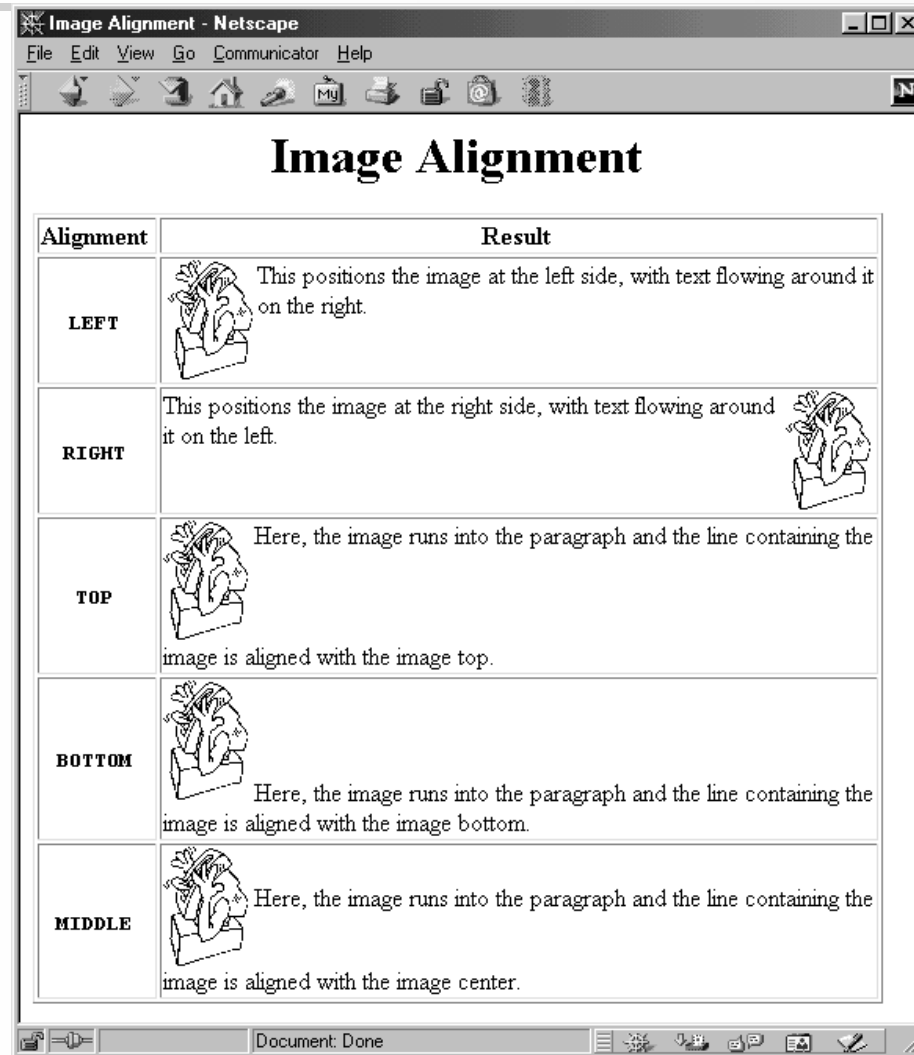
- Attributes:

- SRC (required)
- ALT (required for xhtml)
- ALIGN (see <br clear="ALL" />)
- WIDTH, HEIGHT
- HSPACE, VSPACE
- BORDER
- USEMAP, ISMAP

Image Alignment, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD><TITLE>Image Alignment</TITLE></HEAD>
<BODY>
<H1 ALIGN="CENTER">Image Alignment</H1>
<TABLE BORDER=1>
  <TR><TH>Alignment
    <TH>Result
  <TR><TH><CODE>LEFT</CODE>
    <TD><IMG SRC="rude-pc.gif" ALIGN="LEFT"
      ALT="Rude PC" WIDTH=54 HEIGHT=77>
      This positions the image at the left side,
      with text flowing around it on the right.
  <TR><TH><CODE>RIGHT</CODE>
    <TD><IMG SRC="rude-pc.gif" ALIGN="RIGHT"
      ALT="Rude PC" WIDTH=54 HEIGHT=77>
      This positions the image at the right side,
      with text flowing around it on the left.
  ...
</TABLE>
</BODY>
</HTML>
```

Image Alignment, Result



http://triton.towson.edu/~dgregory/image_alignment.html/

Image Maps

- Contain more than one hyperlink
- Different areas of the image can be clicked on
- Server-Side Image Map

```
<a href="/cgi-bin/imagemap/image1.map">  
    
</a>
```
- Client-Side Image Map
 - All processing on the client

Client-Side Image Maps

- Client-side Image Maps have 3 components:
 - A graphic file (.gif, .jpeg or .jpg, or .png)
 - A map definition delimited by `<map>` tags containing the pixel coordinates and URL information for each area.
 - An `` tag that contains the `usemap` attribute that indicates which map to reference.

HTML Frames

- Advantages and disadvantages of frames
- FRAME template
- Defining rows and cols in a FRAMESET
- Common FRAME and FRAMESET attributes
- Nested frames
- Targeting a document to a named FRAME cell

HTML Frames

- Frames divide browser window into smaller rectangular cells, or sub-windows, each displaying a different HTML document.
- Frames provide advanced navigational control.
- Links within frames specify where in the frameset the pages should load, or if they should break out of the frameset and load in a new, full size browser window.

Frame Advantages

- Certain parts of the interface (e.g., a TOC) are always on the screen
- Can avoid retyping common sections of multiple Web pages
- Consistent use across a large site sometimes simplifies user navigation
- A convenient way to mix text-oriented HTML with Java applets
- Image maps are more convenient if the map image remains on screen and only the results section changes

Frame Disadvantages

- The meaning of the “Back” and “Forward” buttons can be confusing
- Poorly designed frames can get the user lost
- Hard to find real URL of a page you want
 - Printing problems!
- Hard to bookmark "configuration"
- Some very old browsers do not support frames
- Security
 - Hackers can insert frame cells into your pages in some circumstances, perhaps stealing information intended for your site

Frame Template

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Frameset//EN">
<HTML>
<HEAD><TITLE>Document Title</TITLE></HEAD>

<FRAMESET ...>
  <!-- FRAME and Nested FRAMESET Entries -->
  <NOFRAMES>
    <BODY>
      <!-- Stuff for non-Frames browsers -->
    </BODY>
  </NOFRAMES>
</FRAMESET>
</HTML>
```

FRAMESET Attributes

- COLS, ROWS
 - A comma-separated list of pixel values, percentages, and weighted remainders
 - FRAMESET entries should *always* specify at least two rows or columns. Netscape problems if not!
 - Examples

```
<FRAMESET COLS="25% , * , *">
```

```
...
```

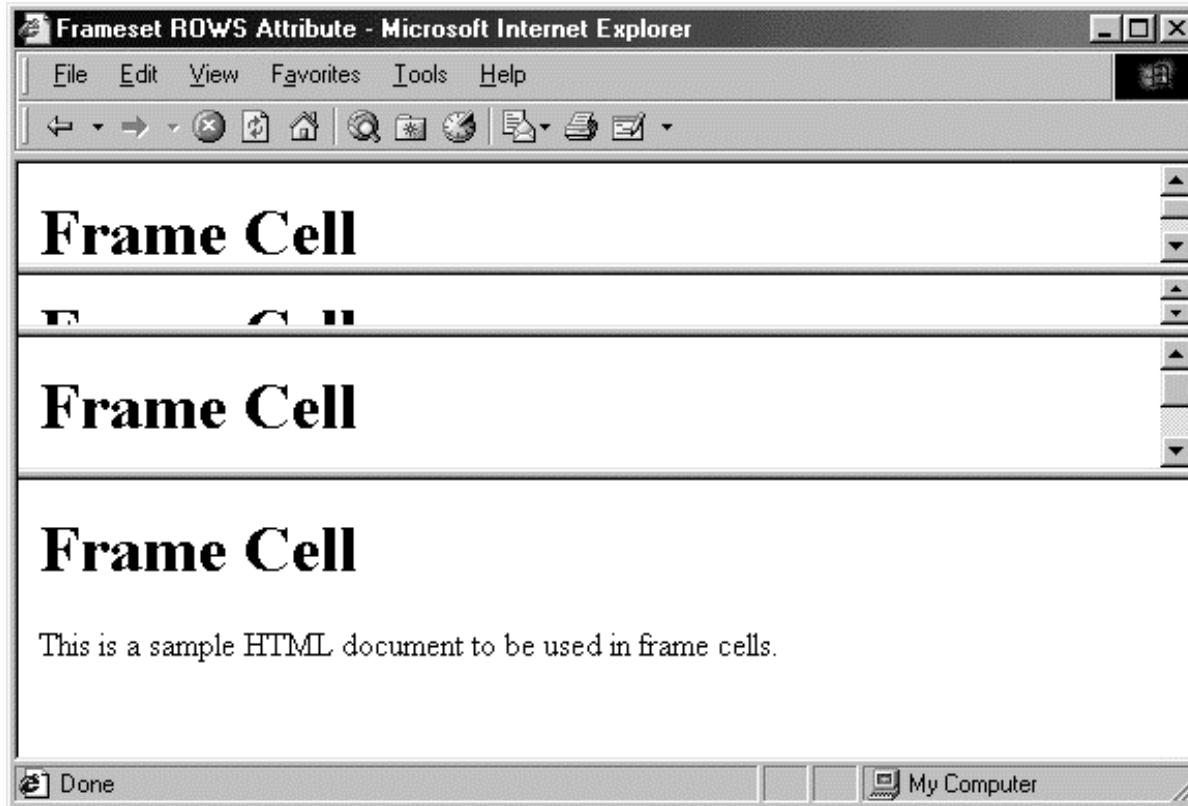
```
</FRAMESET>
```

```
<FRAMESET ROWS="50 , 10% , * , 2*">
```

```
...
```

```
</FRAMESET>
```

FRAMESET ROWS, Example



```
<FRAMESET ROWS="50,10%,*,2*">
```

...

```
</FRAMESET>
```

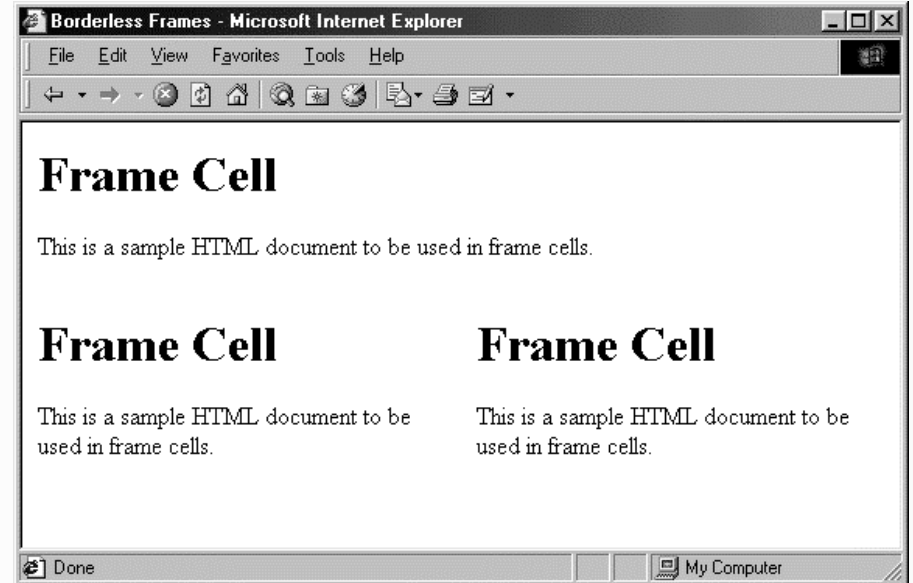
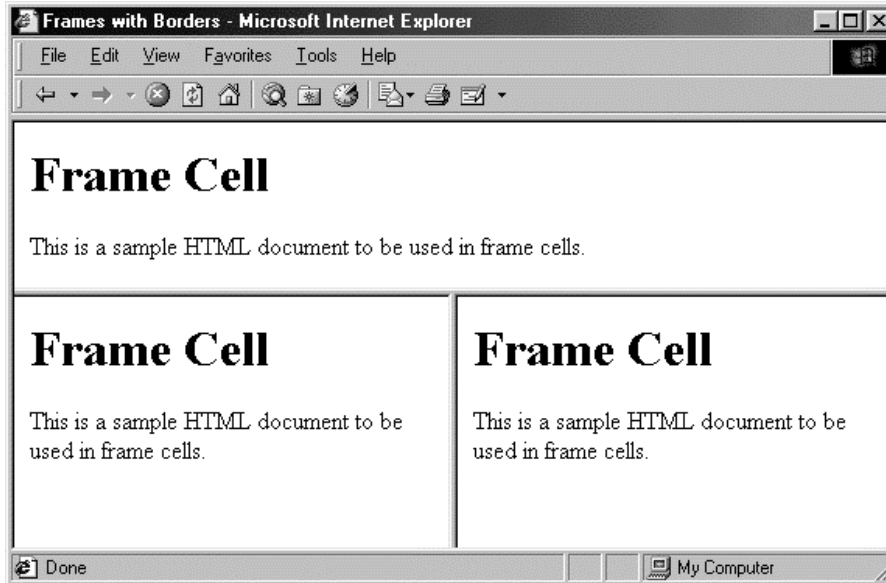
FRAMESET Attributes (Continued)

- FRAMEBORDER
 - Indicates whether borders will be drawn *between* frame cells
 - YES or 1 specifies borders; NO or 0 specifies no border
 - Can be overridden by FRAMEBORDER settings in individual FRAME entries
 - Often used in conjunction with BORDER=0 and FRAMESPACING=0
- BORDER (Netscape), FRAMESPACING (IE)
 - Specify the thickness of the border between cells
 - Apply to outermost FRAMESET only

FRAMESET Attributes (Continued)

- **BORDERCOLOR**
 - Sets the color of the border between cell, using either a hex RGB value or color name

Frame Border, Examples



FRAME: Specifying Content of Frame Cells

- SRC
 - URL of the document to place in the frame cell
- NAME
 - Supplies destination for TARGET attribute of hypertext links
- FRAMEBORDER, BORDERCOLOR
- MARGINWIDTH, MARGINHEIGHT
 - Specifies the left/right and top/bottom cell margins, respectively
- SCROLLING
 - Indicates whether cells should have scrollbars
- NORESIZE
 - Disables the ability to resize the frame cells

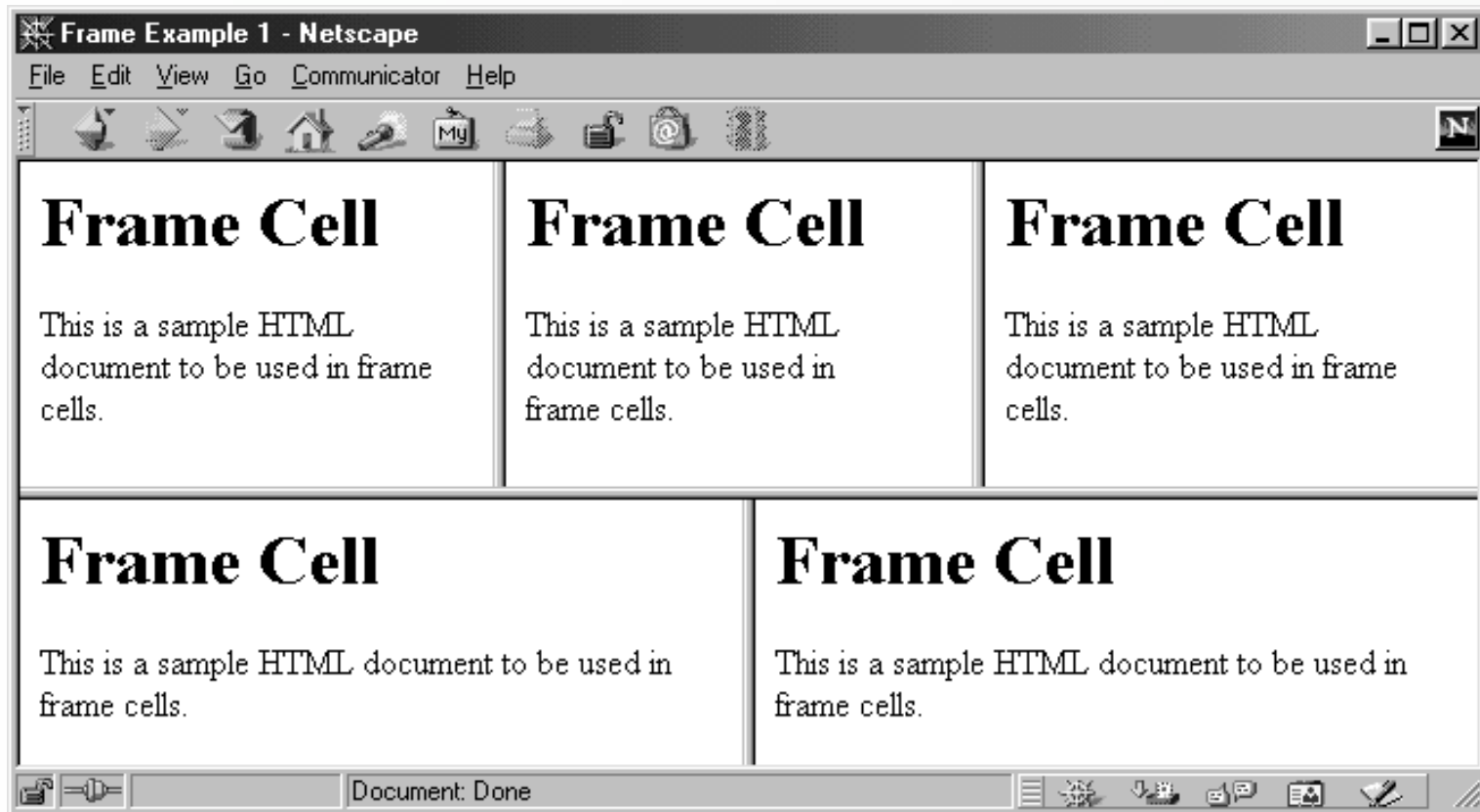
Frame Example 1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD><TITLE>Frame Example 1</TITLE></HEAD>
<FRAMESET ROWS="55%,45%">
  <FRAMESET COLS="*,*,*">
    <FRAME SRC="Frame-Cell.html">
    <FRAME SRC="Frame-Cell.html">
    <FRAME SRC="Frame-Cell.html">
  </FRAMESET>

  <FRAMESET COLS="*,*">
    <FRAME SRC="Frame-Cell.html">
    <FRAME SRC="Frame-Cell.html">
  </FRAMESET>

<NOFRAMES>
  <BODY>
    Your browser does not support frames. Please see
    <A HREF="Frame-Cell.html">non-frames version</A>.
  </BODY>
</NOFRAMES>
</FRAMESET>
</HTML>
```

Frame Example 1, Result



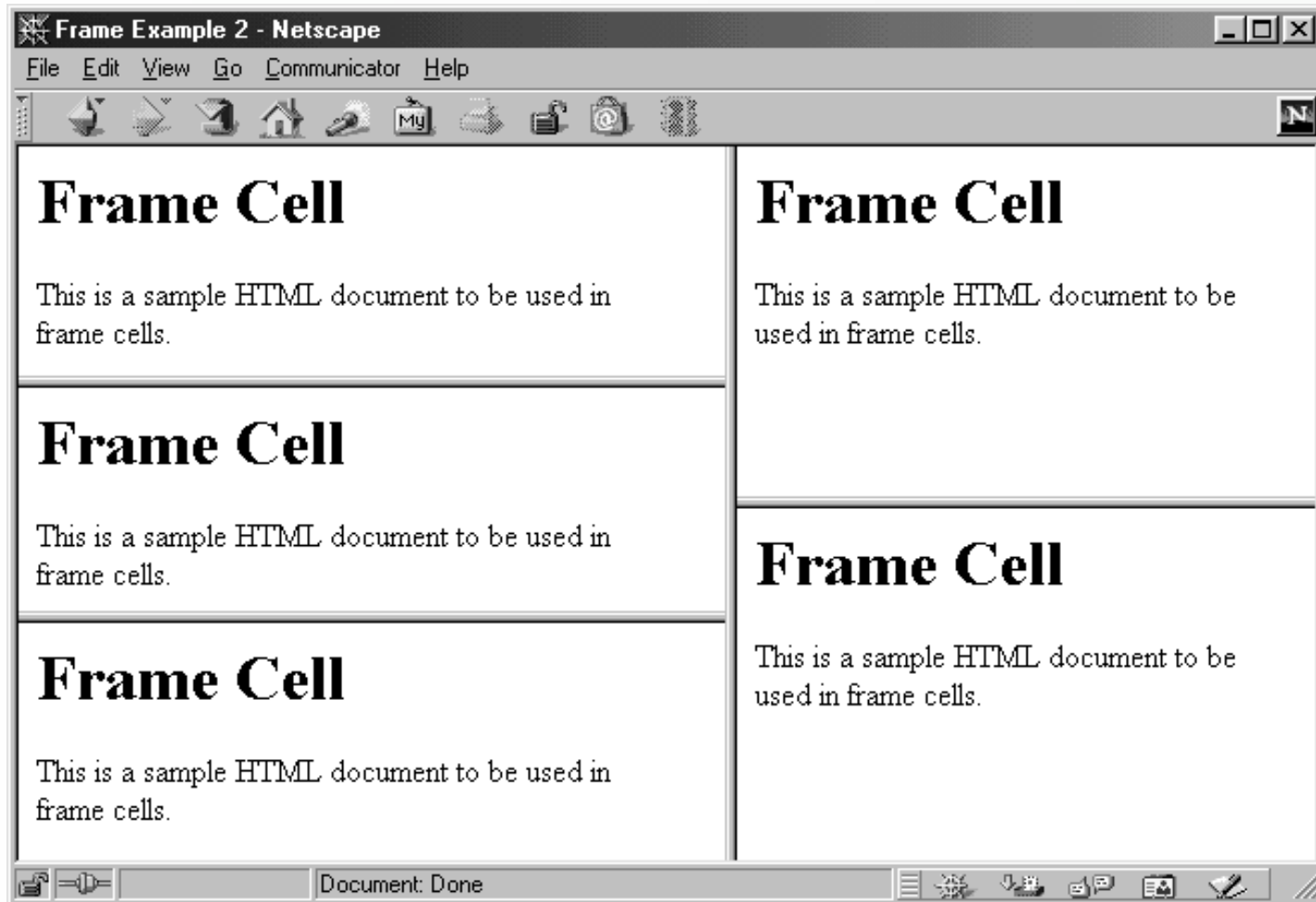
Frame Example 2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD><TITLE>Frame Example 2</TITLE></HEAD>

<FRAMESET COLS="55%,45%">
  <FRAMESET ROWS="*,*,*">
    <FRAME SRC="Frame-Cell.html">
    <FRAME SRC="Frame-Cell.html">
    <FRAME SRC="Frame-Cell.html">
  </FRAMESET>

  <FRAMESET ROWS="*,*">
    <FRAME SRC="Frame-Cell.html">
    <FRAME SRC="Frame-Cell.html">
  </FRAMESET>
</NOFRAMES>
<BODY>
  Your browser does not support frames. Please see
  <A HREF="Frame-Cell.html">nonframes version</A>.
</BODY>
</NOFRAMES>
</FRAMESET>
</HTML>
```

Frame Example 2 Result



Targeting Frame Cells

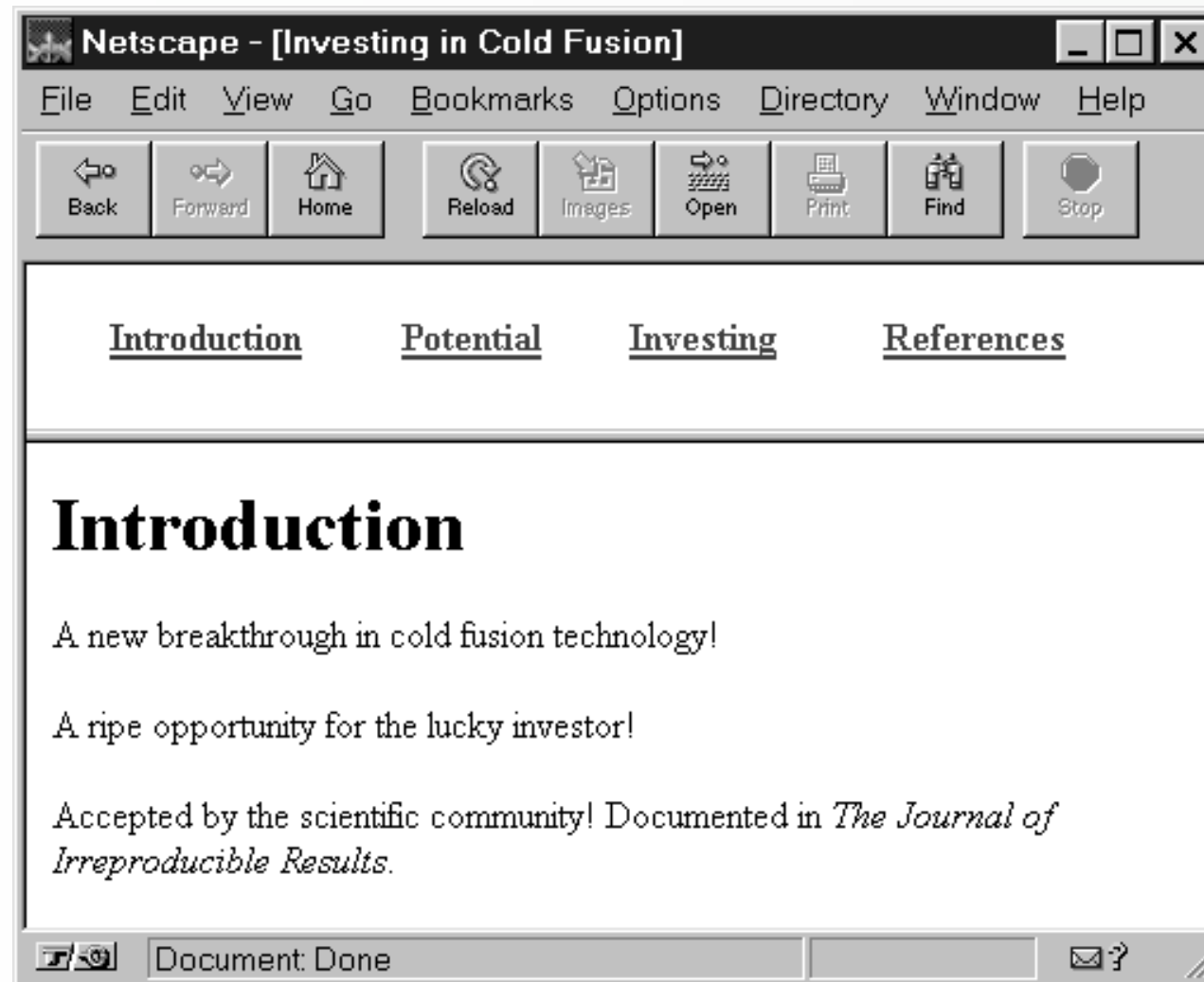
- Specify the cell in which to place a page referenced by a hyperlink
- The NAME Attribute of FRAME

```
<FRAME SRC="..." NAME="cellName1">
```

- The TARGET Attribute of A HREF

```
<A HREF="..." TARGET="cellName">
```

Targeting Example



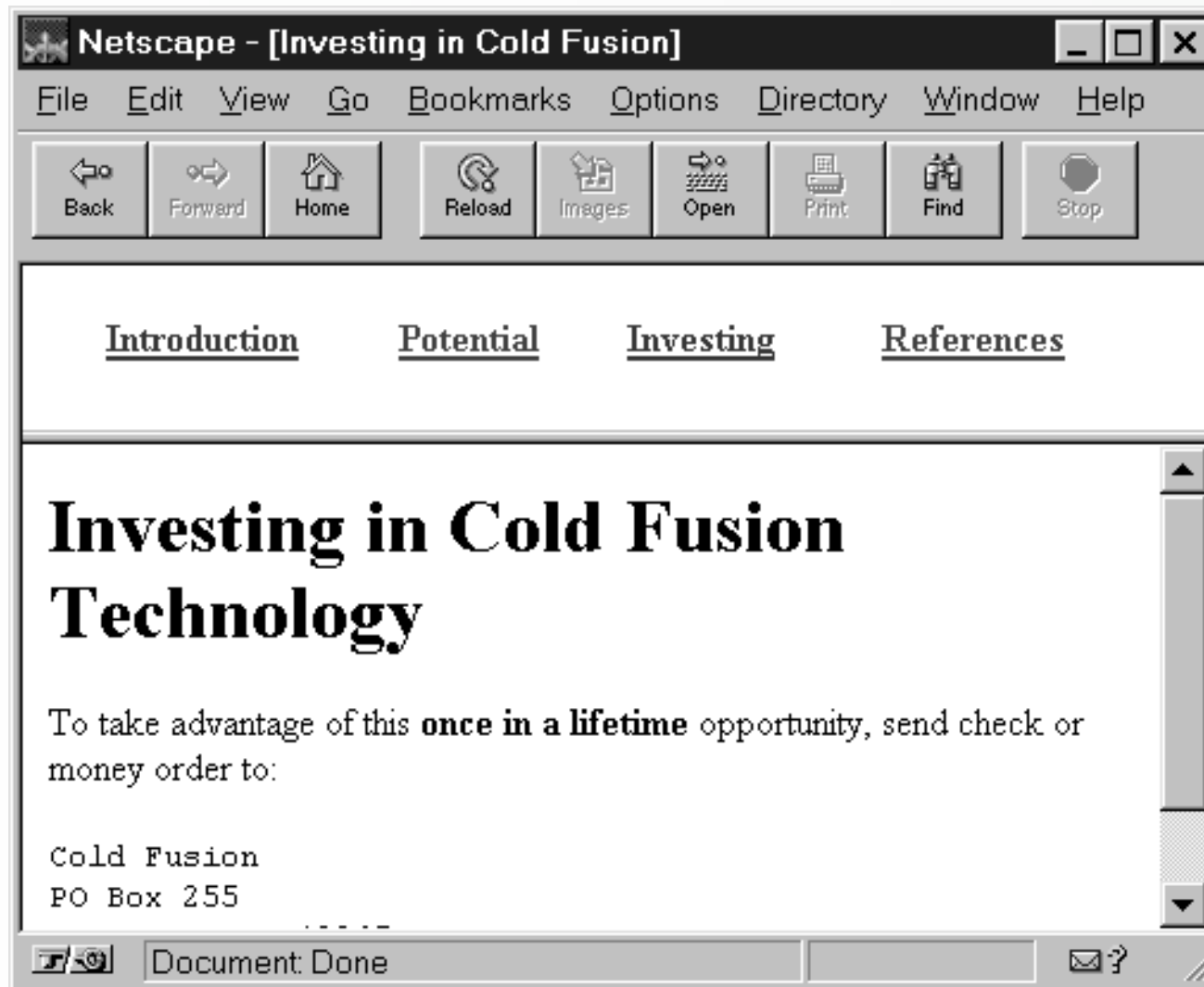
Cold-Fusion.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
  <TITLE>Investing in Cold Fusion</TITLE>
</HEAD>
<FRAMESET ROWS="75, *">
  <FRAME SRC="TOC.html" NAME="TOC">
  <FRAME SRC="Introduction.html" NAME="Main">
<NOFRAMES>
  <BODY>
    This page requires Frames. For a non-Frames version,
    <A HREF="Introduction.html">the introduction</A>.
  </BODY>
</NOFRAMES>
</FRAMESET>
</HTML>
```

TOC.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Table of Contents</TITLE>
</HEAD>
<BODY>
<TABLE WIDTH="100%">
  <TR><TH><A HREF="Introduction.html" TARGET="Main">
    Introduction</A></TH>
  <TH><A HREF="Potential.html" TARGET="Main">
    Potential</A></TH>
  <TH><A HREF="Investing.html" TARGET="Main">
    Investing</A></TH>
  <TH><A HREF="References.html" TARGET="Main">
    References</A></TH></TR>
</TABLE>
</BODY>
</HTML>
```

Targeting Example Results



Predefined Frame Names

- `_blank`
 - Load document into a new browser window
- `_top`
 - Causes the linked document to take up the whole browser window
 - Document will not be contained in a frame cell
- `_parent`
 - Places document in the *immediate* FRAMESET parent
 - Same as `_top` if no nested frames
- `_self`
 - Place document in current cell
 - Only necessary to override a BASE entry

Frames Summary

- Frames require a Frameset DOCTYPE for validation
- A FRAMESET can be divided either into columns or rows
 - To create both rows *and* columns use nested FRAMESETs
- By giving a FRAME a name, documents can be targeted to the named frame cell
 - `<FRAME ... NAME="...">`
 - ``
- There are four predefined frame names
 - `_blank`, `_top`, `_parent`, and `_self`

HTML Forms

- Sending data from forms
- The FORM element
- Text controls
- Push buttons
- Check boxes and radio buttons
- Combo boxes and list boxes
- File upload controls
- Server-side image maps
- Hidden fields
- Grouping controls
- Tab ordering

HTML Forms

- Sending Data to Server-Side Programs
 - Forms are a means for user-supplied data to be passed on to the server.
- Forms are supported by almost all browsers.
- `<form>` tags are required to give an **action** attribute that gives a URL of the application to receive and process the form data

```
<form action="http://www.xyz.com/cgi-bin/submit">
```

```
    form elements
```

```
</form>
```

method Attribute

- The second required attribute of the `<form>` tag is the `method` attribute with a value of `GET` or `POST`, e.g.,

```
<form method="GET" action="...">
```

id (name) Attribute

- Use the id attribute only if plan to manipulate the elements of a form with JavaScript, e.g.,

```
<form method="GET" action="..."  
id="form1" name="form1">  
    form elements  
</form>
```

Form Data Encoding

- Form data can be encoded depending on the processing to be done on the server side.
- By default, the data is encoded in a format referred to as:
application/x-www-form-urlencoded.
- Encoding translates spaces and other characters not permitted in URLs into their hexadecimal equivalents.

Form Input Elements

- Most form elements are created with the `<input>` tag.
- The only required attributes for all elements are **type** and **name**, e.g.,

```
<input type="text" name="address"
size="30" maxlength="256" />
```
- Element types
 - text field
 - text area
 - checkboxes
 - radio buttons
 - pop-up menu

Text Fields

- Conventional fields
- Masked fields (for secure data)
- Fields naming a file to be transmitted as part of form data
- Cannot restrict kind of data entered, just length

Multi-line Text Areas

- Upon submission to the server, each line separated by “%0D%0A”.
- Can insert text within a `<textarea>` tag as default text:

```
<textarea name="address" cols="40" rows="3">  
  name text  
  address text  
  address text  
</textarea>
```
- The text is automatically scrolled if it exceeds the dimensions of the text area.

Wrap Attribute

- If the **wrap** attribute is set to **virtual**, text wrapped in the text area, but sent with carriage returns only where user hits [Enter] key.
- If the **wrap** attribute is set to **physical**, then text wrapped and sent to server as if user hit [Enter] after each line.

Misc Form Items

- Hidden Fields

- Can be used to hide form info

```
<input type="hidden" name="action"  
value="formversionvalue='ver 1.1'" />
```

- Tab order of fields

- Same order as elements listed in **<form>** section
- Can exclude element using **notab** attribute
- Can use **taborder** attribute to explicitly specify order

Action Buttons

- Cause immediate action on the whole form, and not just a single field
- Can have more than one in a form
- Types:
 - Submit
 - Reset
 - Regular
 - Clickable image

Submit Button

- When clicked, submits all form data to server
- When name/value fields omitted, displays a button with name "Submit"

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

- If value given, labels button with string given in the value field

```
<input type="submit" value="BUY NOW!" />
```

- If name/value given, value attribute is added to the parameter list sent to the server, so can determine which button pressed

```
<input type="submit" name="Ship-Style"  
value="Ship Overnight" />
```

Reset Button

- `<input type="reset" />`
- Does not initiate processing, but resets the form elements.
- Server is not notified when the button is selected.
- By default, creates a button labeled “Reset”.

Regular Button

- Only useful when using JavaScript
- When have **onclick** attribute, can have JavaScript code executed when pressed
- Could be used to:
 - validate form contents (**NOT WITH SUBMIT!**)
 - update fields
 - etc.
- `<input type="button" value="Check Values" onclick="validateform()" />`

Clickable Image Button

- Acts as a submit button
- `<input type="image" src="images/image2.gif" />`

JavaScript Form Event Handlers

- **onclick**
 - When checkbox, radio, reset, or submit button clicked
- **onselect**
 - When user selects a text input box or text area
- **onfocus**
 - When user moves mouse over text input box or text area
- **onblur**
 - When user moves away from a text input box or text area
- **onchange**
 - When user changes value of text input box or text area

GET and POST

- **GET**

- Form data is sent in one transmission, with the form data following a “?” after the URL.

- **POST**

- Form data is sent in two transmissions. First, the server is contacted, and then the data is sent in a second transmission.

GET versus POST

- For best performance, send data using **GET**.
- If server operating system limits the length of command lines, use **POST**.
- For inexperienced users, **GET** is easier.
- **POST** provides better security.
- **GET** is the default if not specified.

Sending Data with GET

...

```
<h2>A Sample Form Using GET</h2>
```

```
<form method="GET"
```

```
    action="http://localhost:8088/SomeProgram">
```

```
  <p>
```

```
    First name:
```

```
    <input type="text" name="firstName" value="Joe" />
```

```
    <br />
```

```
    Last name:
```

```
    <input type="text" name="lastName" value="Hacker" />
```

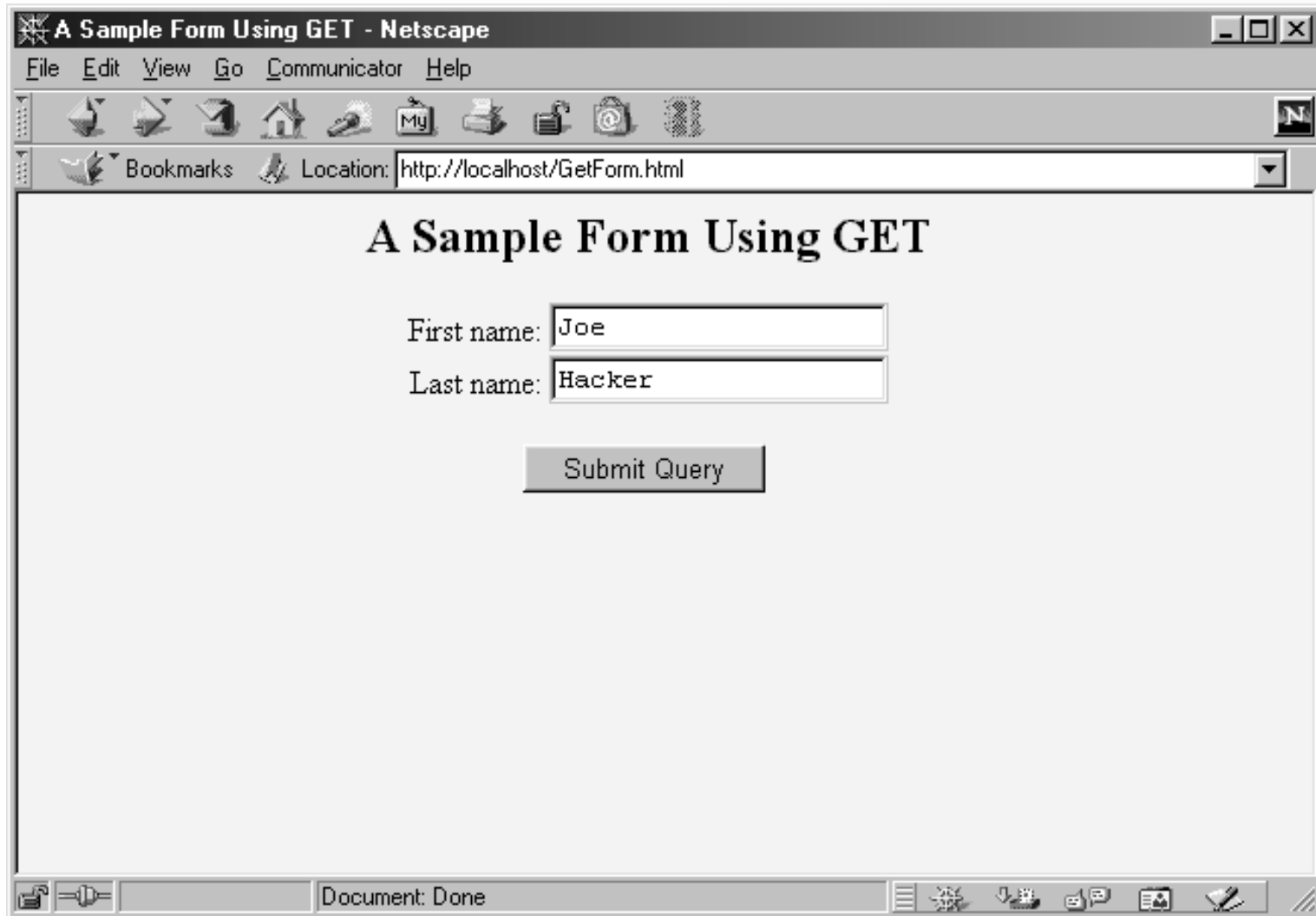
```
    <br />
```

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

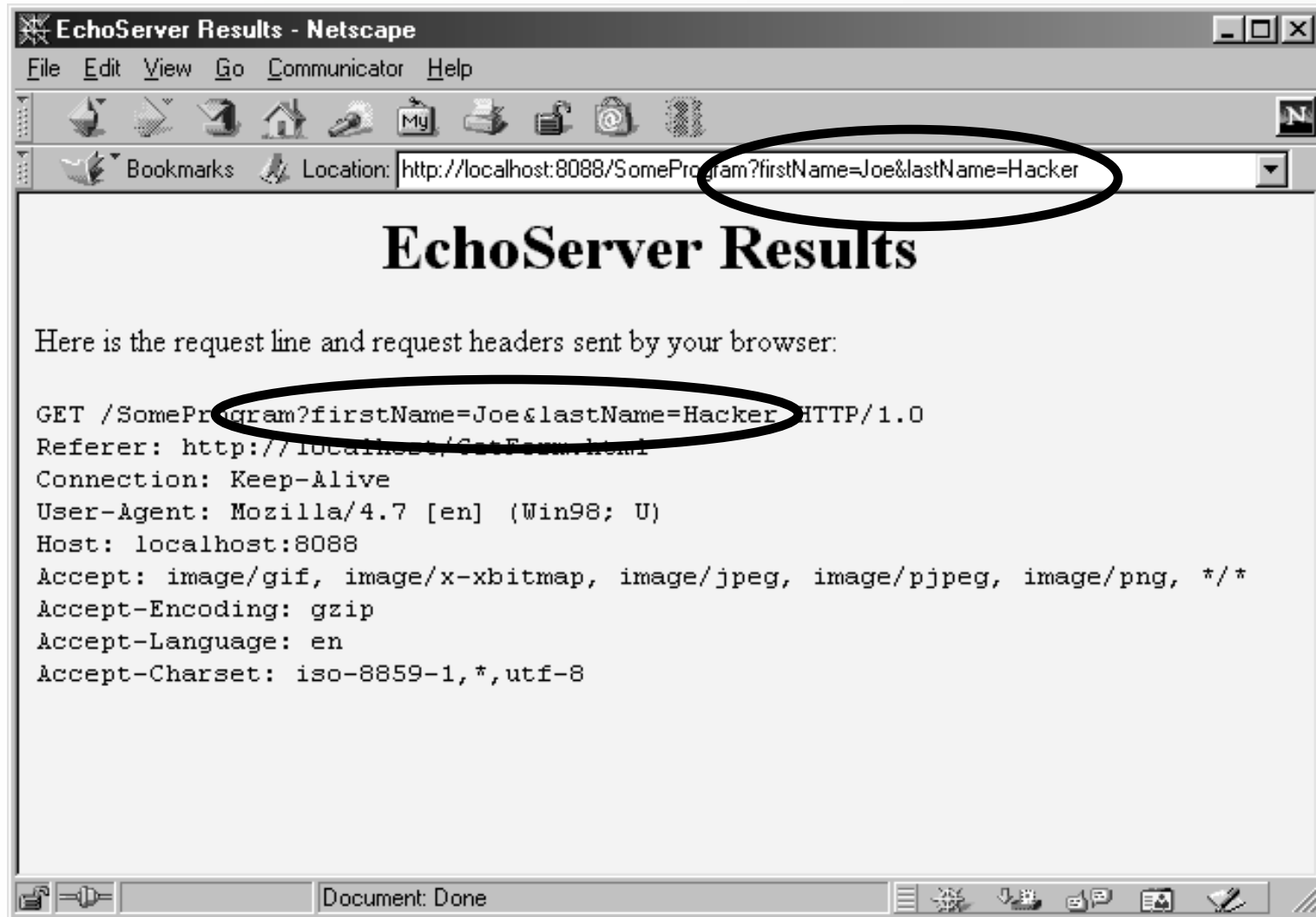
```
  </p>
```

```
</form>
```

Initial Result



Submission Result

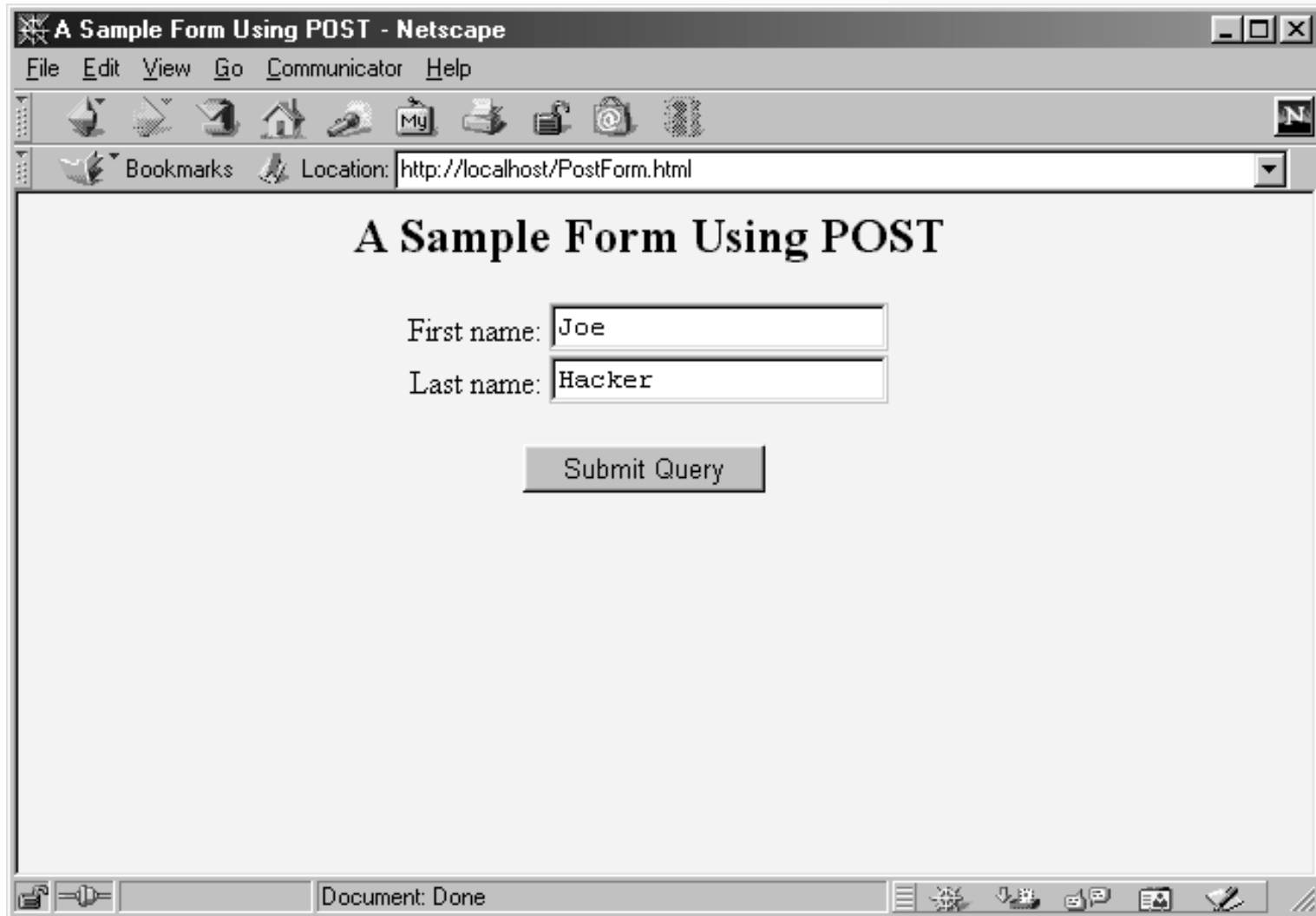


Sending Data with POST

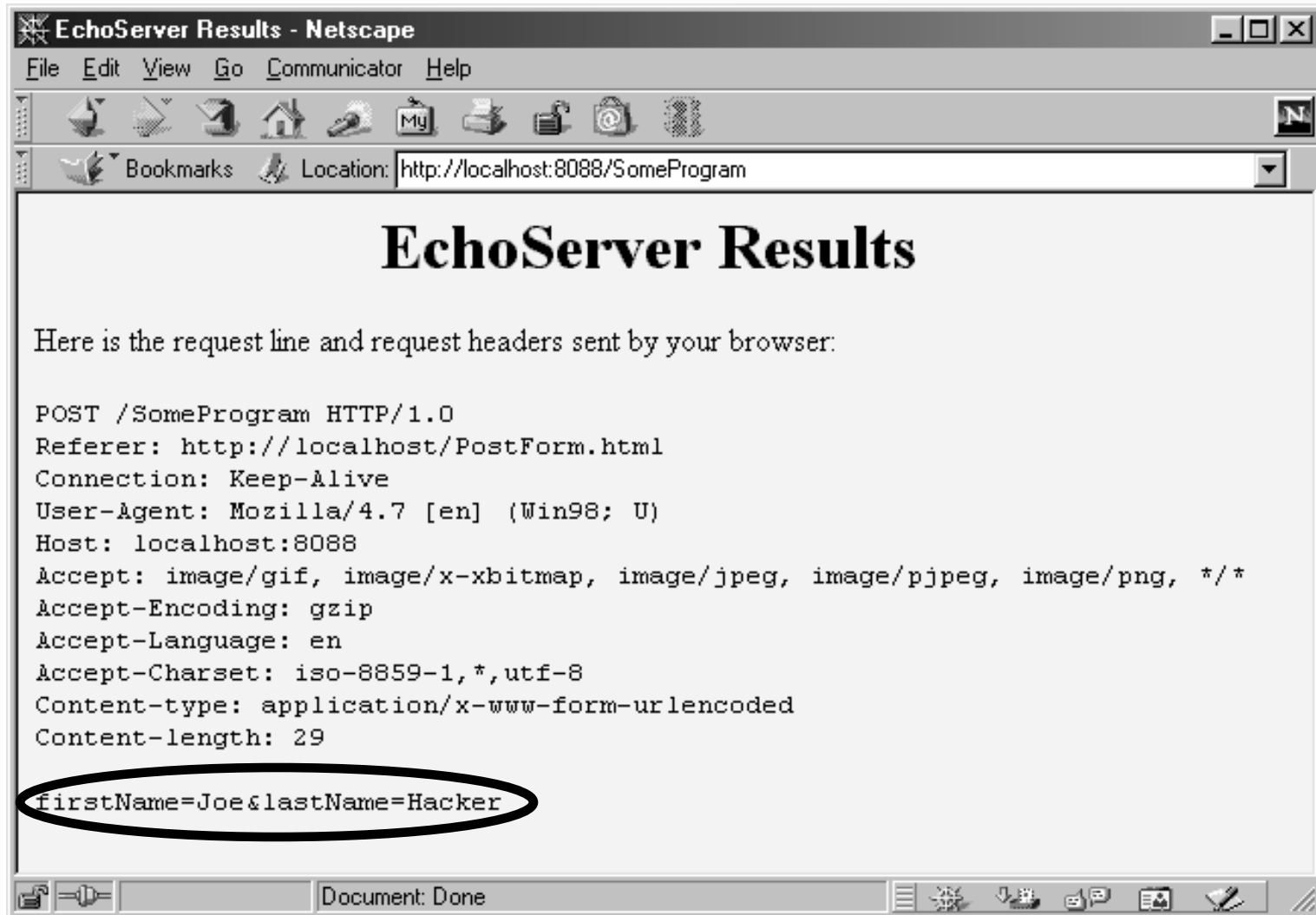
```
...
<h2>A Sample Form Using POST</h2>

<form method="POST"
      action="http://localhost:8088/SomeProgram">
  <p>
    First name:
    <input type="text" name="firstName" value="Joe" />
    <br />
    Last name:
    <input type="text" name="lastName" value="Hacker" />
    <br />
    <input type="submit" />
  </p>
</form>
```

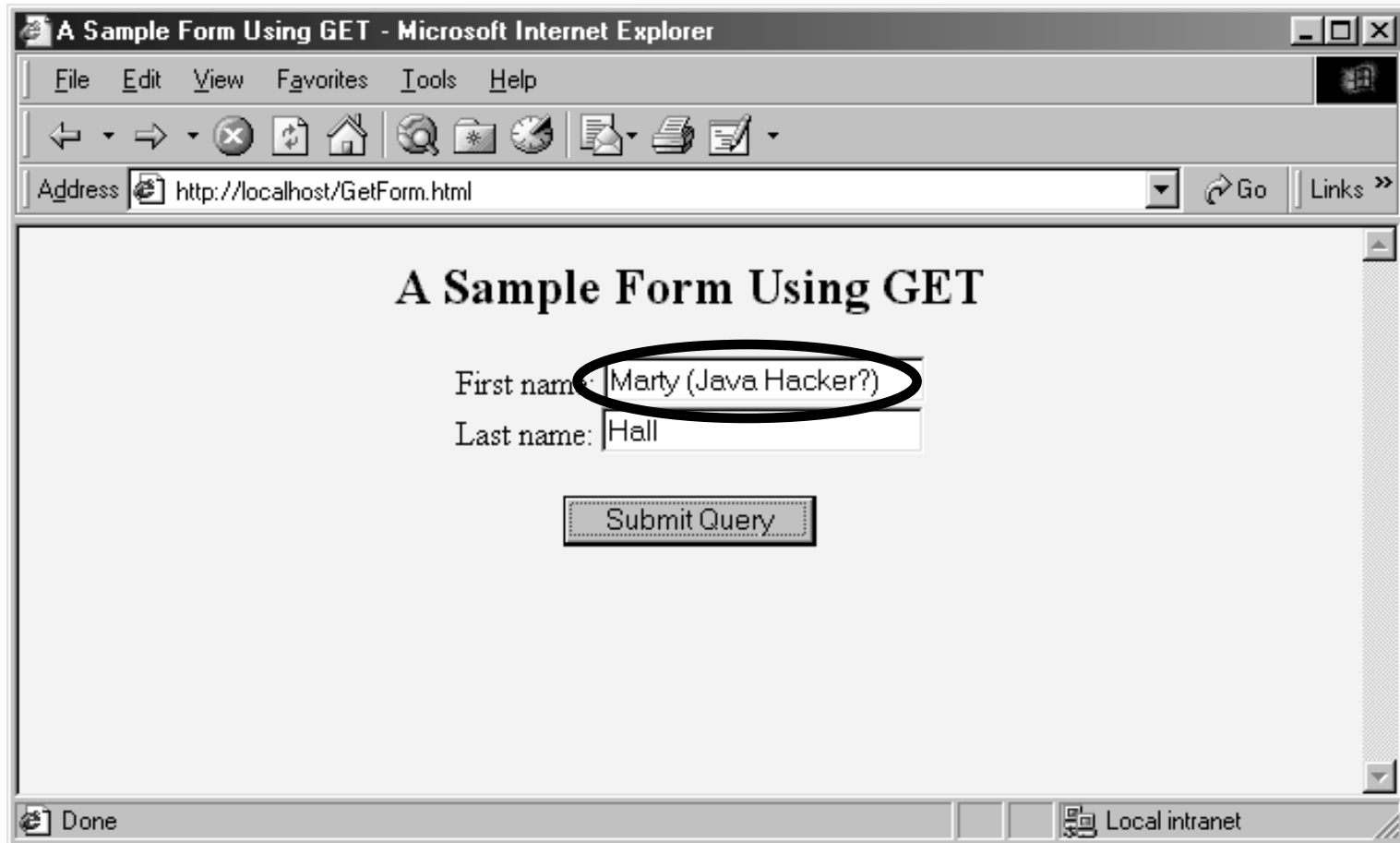
Initial Result



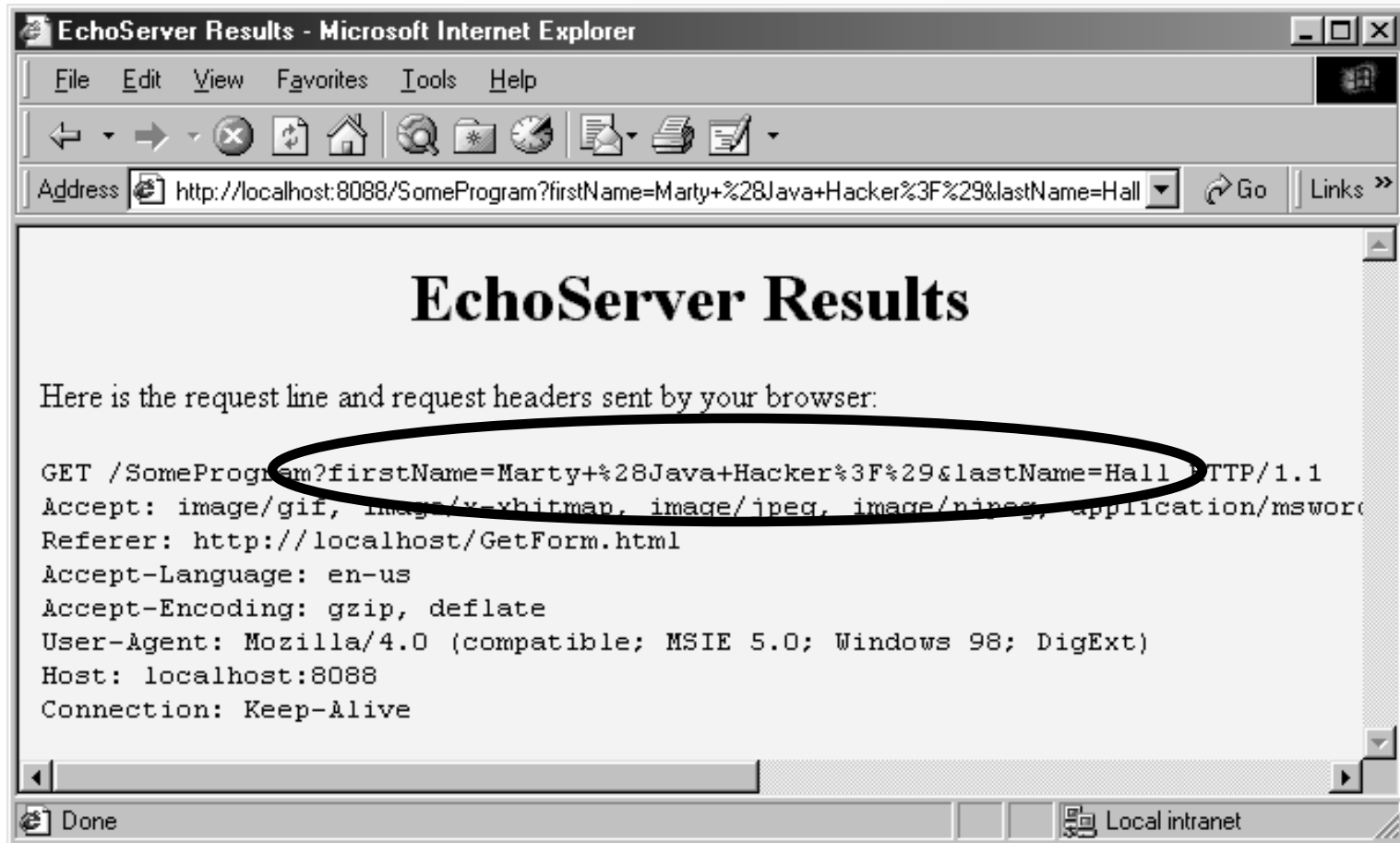
Submission Result



URL Encoding: Original Form



URL Encoding: Result



Text Controls

- Textfields

- `<input type="text" id="..." ...>`
 - `VALUE` can give original value

- Password Fields

- `<input type="password" name="..." ...>`
 - *Always* use `POST`



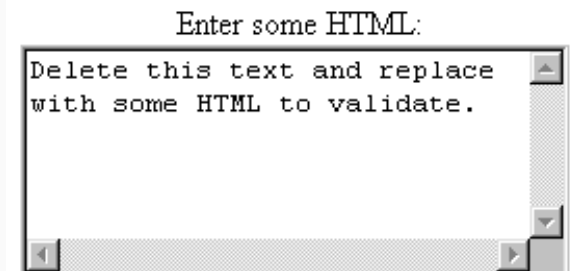
- Text Areas

- `<textarea name="..."`
`rows="..." cols="...">`

...

`</textarea>`

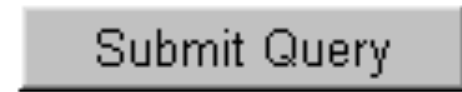
- Interpretation of regular HTML tags turned off between `<TEXTAREA...>` and `</TEXTAREA>`



Push Buttons

- Submit Buttons

- `<input type="submit" ...>`
 - Use `name` if you have multiple buttons
 - Use `value` to change button's label



- Reset Buttons

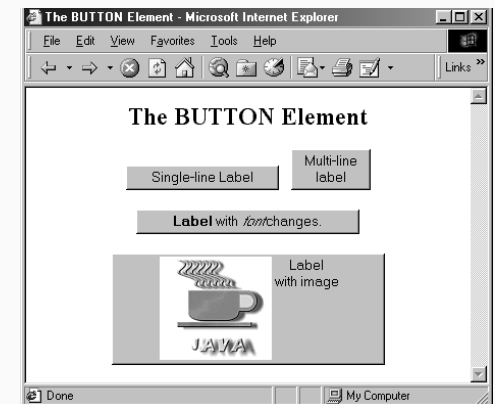
- `<input type="reset" ...>`
 - Use `value` to change button's label

- JavaScript Buttons

- `<input type="button" onclick="someJavaScriptFunction()" ...>`

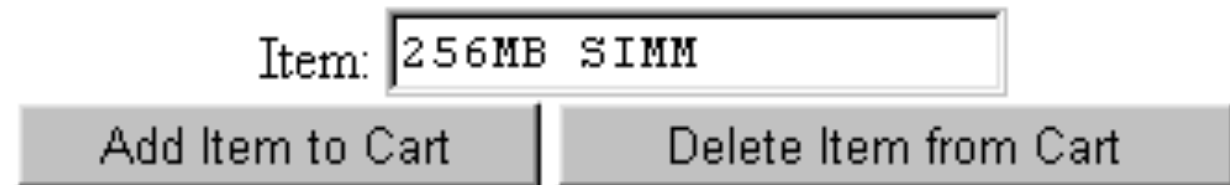
- Fancy Buttons

- `<button type="submit">HTML</button>`
 - Internet Explorer and Netscape 6 only



Using Multiple Submit Buttons

```
<p>
Item:
<input type="text" name="Item" value="256MB SIMM"
  />
<br />
<input type="submit" name="Add"
  value="Add Item to Cart" />
<input type="submit" name="Delete"
  value="Delete Item from Cart" />
</p>
```



Item:

Check Boxes

- Format

- `<input type="checkbox" name="..." ... />`

- The `CHECKED` attribute makes it initially checked
 - Name/value pair sent only if checkbox is checked when form is submitted

- Example code

- ```
<p><input type="checkbox" name="noEmail"
 checked="checked" />
```

- ```
Check here if you do <i>not</i> want to
get our email newsletter </p>
```

- Example result

- ```
 Check here if you do not want to get our email newsletter
```

# Radio Buttons

- Format

- `<input type="radio" name="..." value="..."... />`

- All radio buttons in a group should have same NAME
    - Only one button in a group can be pressed; pressing a different one causes previous one to pop out

- Example

```
<dl>
 <dt>Credit Card:</dt>
 <dd><input type="radio" name="creditCard"
 value="visa" />
 Visa</dd>
 <dd><input type="radio" name="creditCard"
 value="mastercard" />
 Master Card</dd>
 ...
</dl>
```

Credit Card:

- Visa
- Master Card
- Java Smart Card
- American Express
- Discover

# Combo Boxes

- Format

- SELECT gives NAME
- OPTION gives VALUE

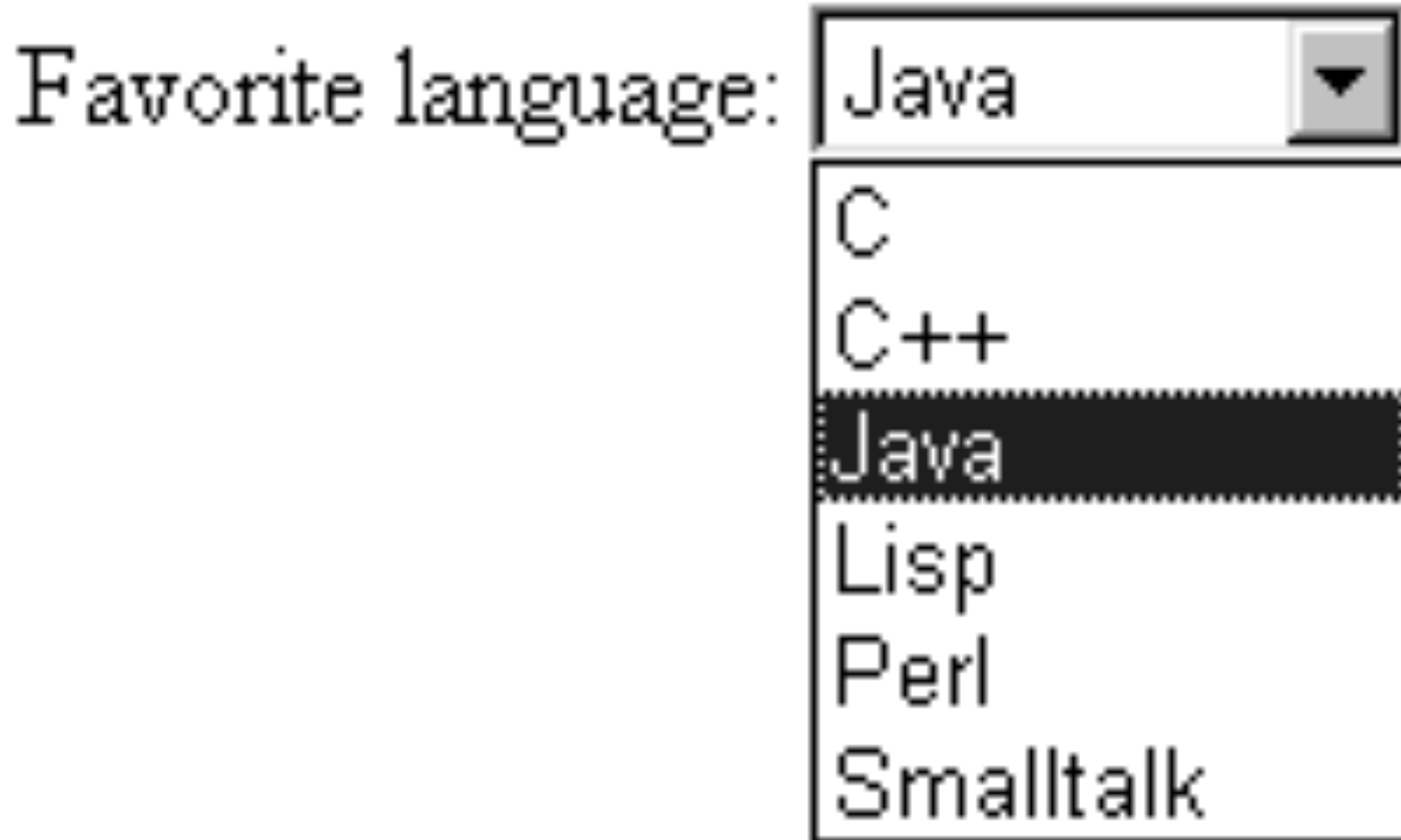
Favorite language:

- Example

```
<p>Favorite language:
<select name="language">
 <option value="c" />C
 <option value="c++" />C++
 <option value="java" selected="selected" />Java
 <option value="lisp" />Lisp
 <option value="perl" />Perl
 <option value="smalltalk" />Smalltalk
</select></p>
```

# Combo Boxes (cont)

---



# List Boxes

- Format
  - Identical to combo boxes
  - specify `MULTIPLE`
- Example

Languages you know:



```
<p>Languages you know:

<select name="language" multiple="multiple">
 <option value="c" />C
 <option value="c++" />C++
 <option value="java" selected="selected" />Java
 <option value="lisp" />Lisp
 <option value="perl" selected="selected" />Perl
 <option value="smalltalk" />Smalltalk
</select></p>
```

# Other Controls and Options

---

- File upload controls
  - Lets user select a file and send it to the server
- Server-side image maps
  - User clicks on an image and form gets submitted.
  - Form data gets sent as *name.x=x-pos&name.y=y-pos*
- Hidden fields
  - Preset `NAME` and `VALUE` sent with form submission..

# Other Controls and Options (Continued)

---

- Grouping Controls
  - `FIELDSET` lets you visually group forms.
  - Internet Explorer and Netscape 6 only.
- Tab order control
  - `TABINDEX` (Internet Explorer and Netscape 6 only)

# HTML Forms Summary

---

- General process
  - FORM uses ACTION to specify base URL
  - Input elements each have a NAME
  - User enters values
  - When form submitted, URL is  
baseURL?name1=value1&name2=value2&...
  - For POST requests, name/value pairs sent on separate line  
(not part of URL)
- Textfields
  - `<input type="text" ... />`
- Submit Buttons
  - `<input type="submit" ... />`

# Questions?

---