

## XHTML

- A newer version of HTML, standardized in 2000
- Describes the *content* and structure of information on a web page
  - Not the same as the *presentation* (appearance on screen)
- Uses a markup format called XML (XML + HTML = XHTML)
- Though the browser will accept some malformed HTML, we'll write "strict" XHTML that complies to the official web standards

## Why use XHTML and web standards?

- It is important to write proper XHTML code and follow proper syntax.
- More rigid and structured language
- More interoperable across different web browsers
- More likely that our pages will display correctly in the future

## XML Syntax

- First line is XML declaration
  - `<?xml version="1.0" encoding="ISO-8859-1" ?>`
  - Nothing can precede this declaration, not even spaces
- All elements must have closing tags
  - `<p>This is a paragraph</p>`
  - `<br />` *Empty element tag*
- Tags are case sensitive
  - `<a>Illegal</A>`
  - `<a>Correct</a>`
- Elements must be properly nested
  - `<a><b>Illegal</a></b>`
  - `<a><b>Correct</b></a>`
- Attribute values must always be quoted
  - `<a sample=illegal>`
  - `<a sample='correct'>`

## XHTML Validation Service

- <http://validator.w3.org>



- Checks your HTML code to make sure it meets the official strict XHTML specifications
- More picky than the web browser, which may render malformed XHTML correctly

## Outline

- Global HTML structure
- Frames
  - Multi-view presentation of documents
- Forms
  - User-input: text fields, buttons, menus, etc.
- XHTML validator

## Document Structure

```
<?xml version=" 1.0" encoding=" UTF-8 " ?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.0 Strict//EN "
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    information about the page
  </head>
  <body>
    page contents
  </body>
</html>
```

## <head>

- *Must* contain a <title> element
  - Used as window titles, in favourite lists, search engine results, ...
  - Choose a good, context-rich one - it's displayed a lot.
    - POOR:
      - <title>Introduction</title>
    - BETTER:
      - <title>Introduction to Bee-Keeping</title>
- May contain
  - <meta>
  - <link>

## <meta>

- Provides meta-information about your page
  - Descriptions
  - Keywords for search engines
  - Refresh rates

```
<meta name="Author" content="Joe Smith">
<meta http-equiv="Expires" content="Tue, 20 Oct 2004 14:25:27 GMT">
<meta http-equiv="refresh" content="5;url=http://www.bjp.com" />
<meta name="keywords" lang="en" content="vacation, sunshine">
```

## <link>

- Relationship information

```
<link rel="Index" href="../index.html"/>
<link rel="Next" href="ch3.html"/>
<link rel="Prev" href="ch1.html"/>
<link rel="Start" href="grtitlepage.html"/>
```

- Site navigation bar icon

```
<link rel="shortcut icon" href="myicon.ico" />
```



## <body>

- Displayable content
  - Text, images, graphics, etc.



- Two types of elements

- **block** elements contain an entire large region of content
  - examples: paragraphs, lists, table cells
  - the browser places a margin of whitespace between block elements for separation
- **inline** elements affect a small amount of content
  - examples: bold text, code fragments, images
  - the browser allows many inline elements to appear on the same line
  - must be nested inside a block element

## Frames

- Allow authors to present documents in multiple views
- May be independent windows or subwindows.
- Multiple views offer designers a way to keep certain information visible, while other views are scrolled or replaced.
- Use with caution!

## Frames cont.

frameset : [frameset|frame]+

frameset attributes

rows	number	-- dimensions of rows --
cols	number	-- dimensions of cols --

frame attributes

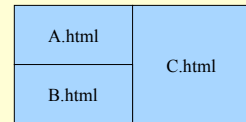
name	string	-- name for targeting --
src	URL	-- source of frame content --
frameborder	(1 0)	1 -- request frame borders? --
scrolling	(yes no auto)	auto -- scrollbar or none -->

## Frames - Size Control

- `<frameset rows="50%, 50%">`
  - divides the space in half
- `<frameset cols="1*, 250, 3*">`
  - middle frame has a width of 250px
  - left gets 25% of remaining, right 75%
- `<frameset rows="30%, 70%" cols="33%, 34%, 33%">`
  - creates a 2x3 grid
- `<frameset rows="30%, 400, *, 2*">`
  - first takes 30%, next takes 400px
  - second last takes 1/3 of remainder, last 2/3.

## Example

```
<?xml version=" 1.0" encoding=" UTF-8 " ?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.0 Frameset//EN "
"www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<head>
  <title>frames example</title>
</head>
<frameset cols="50%, 50%">
  <frameset rows="50%, 50%">
    <frame src="A.html">
    <frame src="B.html">
  </frameset>
  <frame src="C.html">
</frameset>
```



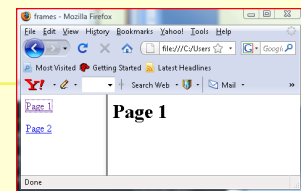
## Frames - Linking Control

- Frames have names
  - `<frame name="foo-frame" />`
  - `<frame name="foo-frame" src="original.html"/>`
- Link can have a "target"
  - `<a href="new.html" target="foo-frame">`
- Pre-defined and reserved targets
  - `_blank` (new target window)
  - `_self` (default)
  - `_parent` (embedded frame sets)
  - `_top` (browser, replace frameset)

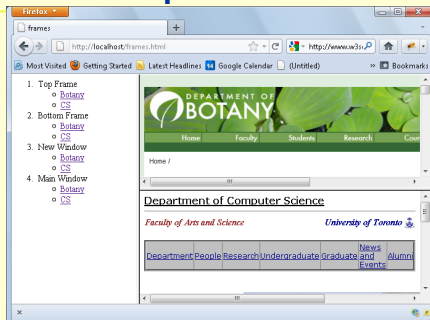
## Example

```
<?xml version="1.0" ?>
<!DOCTYPE html ..... >
<html>
<head><title>frames</title></head>
<frameset cols="30%, 70%">
  <frame src="frameIndex.html"/>
  <frame name="content" />
</frameset>
</html>

<?xml version="1.0" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head><title>Sample Index Frame</title></head>
<body>
  <p><a href="page1.html" target="content">Page 1</a></p>
  <p><a href="page2.html" target="content">Page 2</a></p>
</body>
</html>
```



## Frame Example 2



## Forms

- Gather user input
  - Communicate information to server
- Contain widgets/controls
  - Single and multi-line text boxes
  - Checkboxes
  - Radio buttons
  - Menus
  - Submit and Reset
- Each control has a value
- On submit
  - Encode control values, and send to web server

## <form>

- Contains all controls
- Attributes
  - *action*
    - URL of program to call on server when user presses Submit button
    - e.g., www.cs.toronto/~delara/example.pl
  - *method*
    - Either GET (default) or POST

```
<form method='get' action='someURL'>
```

## Submit and Reset

- Reset
  - Clear all form fields
- Submit
  - Code all input values into a string
  - Ask server to execute application specified in the form's *action* attribute

## <input>

- Specifies text boxes, checkboxes, radio buttons, reset and submit buttons
- Attributes
  - *type*
    - text, checkboxes, radio, reset, submit
  - *name*
  - *value*
    - For checkboxes and radio buttons

## <select>

- Menus
- Attributes
  - *name*
  - *size*
    - 1 – drop down box
    - > 1 – scrolled list
  - *multiple*
    - Allow multiple selection
  - *option*
    - Elements of the list

## Forms Example

Simple form that calls perl script "add.pl" which takes two parameters "A" and "B"

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC ..... >
<html>
<head><title> Simple Calculator </title></head>
<body>
<form action="http://127.0.0.1/cgi-bin/add.pl">
<input type = "text" name = "A" size="3"/> +
<input type = "text" name = "B" size="3"/>
<input type = "submit" value = "Add" /> </form>
</body>
</html>
```

## Forms Example 2

### Chocolates R US

Product Name	Price	Quantity
M&M	\$1.00	<input type="text"/>
Sneakers	\$2.00	<input type="text"/>
Milky Way	\$3.00	<input type="text"/>

Payment Method  Visa  MasterCard

Credit Card Number

Expiration Data Jan 2012

Name