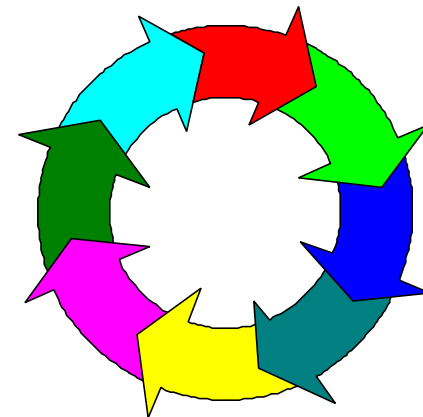
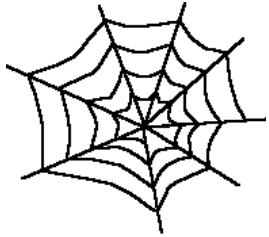


Introduction to Dynamic Web Pages

Ken Coar

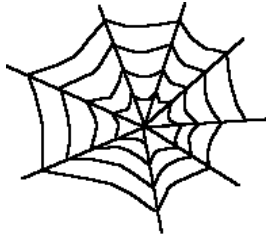
*coar@Apache.Org,
The Apache Group*





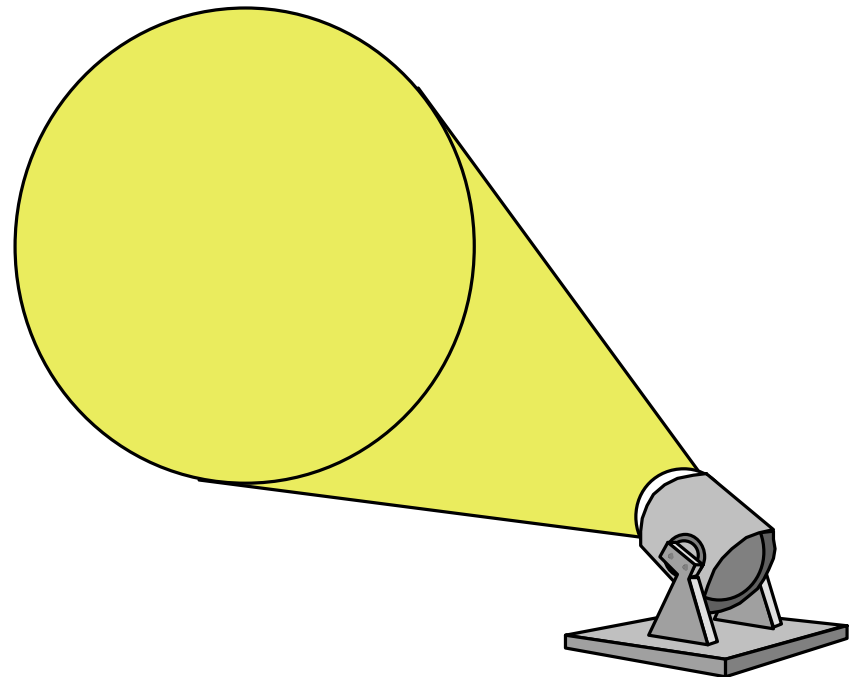
Disclaimers

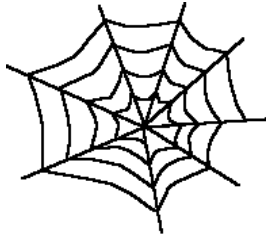
- UNIX is a registered trademark of X/Open Company Ltd
- PostScript is a registered trademark of Adobe Systems, Inc
- All other trademarks are the property of their respective owners, and no misuse is intended



Acknowledgements

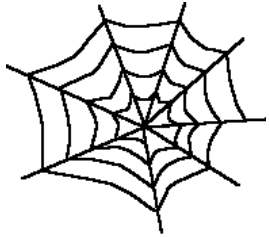
- I would like to gratefully acknowledge the contributions of the following:
 - Neil Gregory, Paul Scherrer Institut





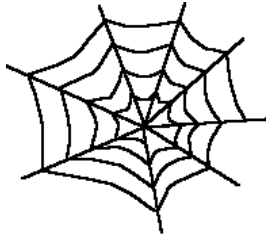
Copies of This Presentation

- *WILL BE* available online at
<http://Web.GoLux.Com/coar/>



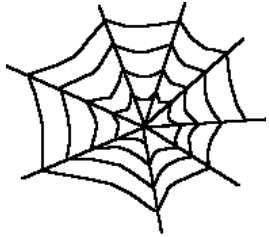
Contents

- What's a dynamic Web page?
- Forms
- Server-side includes
- Server-side scripts
- Client-side scripting
- Java applets



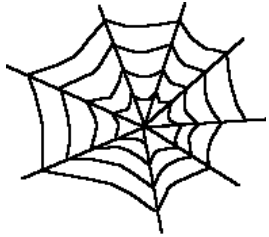
What's a Dynamic Web Page?

- 'Dynamic' means different things to different people
 - Interactive forms
 - Customising page content based on client attributes
 - Reactive pages



Forms

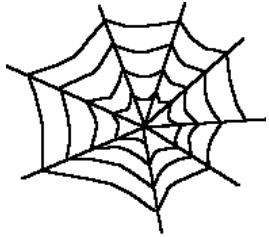
- At base, just a fill-in sheet echoing its paper counterpart
- No client-side data cross-checking or validation; real validation must be done by server upon receipt on a *per-form* basis
- A form *can* be no-action (suitable for viewing or read-only field value propagation), or can specify an URL to which the form contents will be sent *à la* script processing
- Facilities include ability to define default field values, and reset all fields to these defaults at user selection



Forms

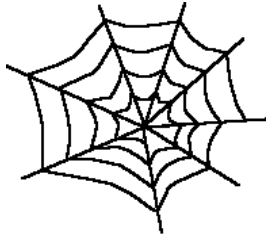
(continued)

- Input types available:
 - Multiple choice (checkboxes)
 - One-of-many selection (radio button)
 - Pick-one from a menu
 - Pick-*N* from a menu
 - Text input (one- or multi-line)
 - Password entry (echoed as ‘*’)
 - Hidden data passing from page to page



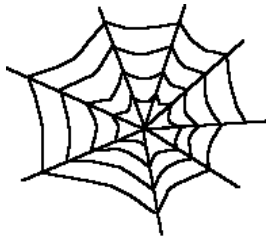
ISINDEX

- Poor man's form -- single field, intended for entering search keywords
- Activated by the `<ISINDEX>` tag
- Shorthand form to accept input and pass it back to the same URL as an argument (in `QUERY_STRING`)
- File/script receives contents of index field as normal script argument input value (GET method)



Server-Side Includes (SSIs)

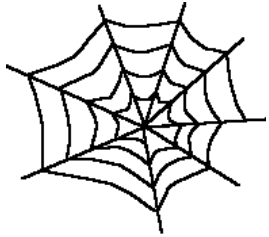
- Directives embedded in Web page for *server* to process before passing page to requestor
- Not all servers support them
- Incurs performance penalty due to parsing overhead
- Generally needs to be enabled by server manager
- Server *may* support *per-directory* enabling



Server-Side Includes (SSIs) (continued)

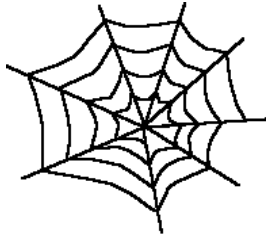
- Typical directives: `exec`, `include`, `echo`
- XSSI (extended SSI) includes conditional directives, such as

```
<!--#if expr="$USER_AGENT = /MSIE/" -->
<P>I see you're using Internet Explorer.</P>
<!--#elseif expr="$USER_AGENT = /Mozilla/" -->
<P>Netscape's Mozilla is your browser.</P>
<!--#else -->
<P>I don't recognise your browser, so I'm
going to pretend it's very dumb.</P>
<!--#endif -->
```



Server-Side Scripts

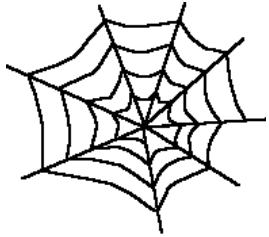
- Applications executed by the server at run-time to process client input or generate document in response to client request
- Some Web servers permit script output to be included in other documents
- Some Web servers can be told to execute scripts when errors (e.g., '404 Not Found') are encountered
- Information about the original request is available to error scripts



Script Arguments

- Arguments can be passed to scripts by appending them to the script URL separated by a question mark
- Available to scripts in all methods
- As it's part of the URL, a particular script response can be bookmarked as long as no POST information was included

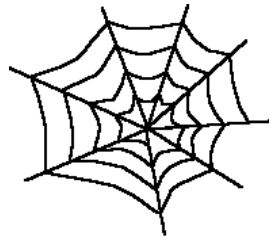




Script Arguments

(continued)

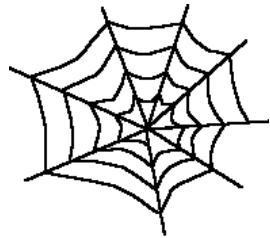
- How the script accesses the arguments depends upon your server interface
- Special characters are encoded (or 'escaped') and need to be converted back. This includes non-printing characters and those with special significance in URLs, such as slashes ('/') and question marks ('?')
- Encoding is done by replacing the character with a '%' and the two-digit hexadecimal representation (e.g., '/' is replaced by '%2f')
- Special case: spaces can be replaced with '+' or '%20'



Path Info - Additional Information for Scripts

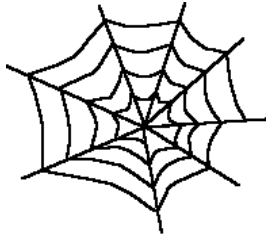
- Whatever's *after* the script-name but *before* the “?*args*”
- Another way to pass information to the script
- As part of the URL, special characters *are* escaped
- Available to scripts in all methods





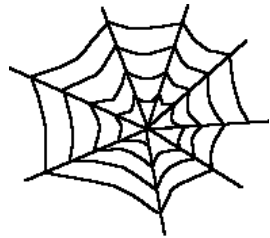
Using Scripts to Handle Errors

- Possible through things like Apache's `ErrorDocument` directive
- Used to prettify and customise error displays, particularly on multi-site hosts
- Use the CGI "`Status:`" response header field to propagate the error to the client, rather than returning the default "`200 OK`" which indicates success -- otherwise any caches will contain your 'successful' error message (which may be due to a temporary condition) rather than nothing at all



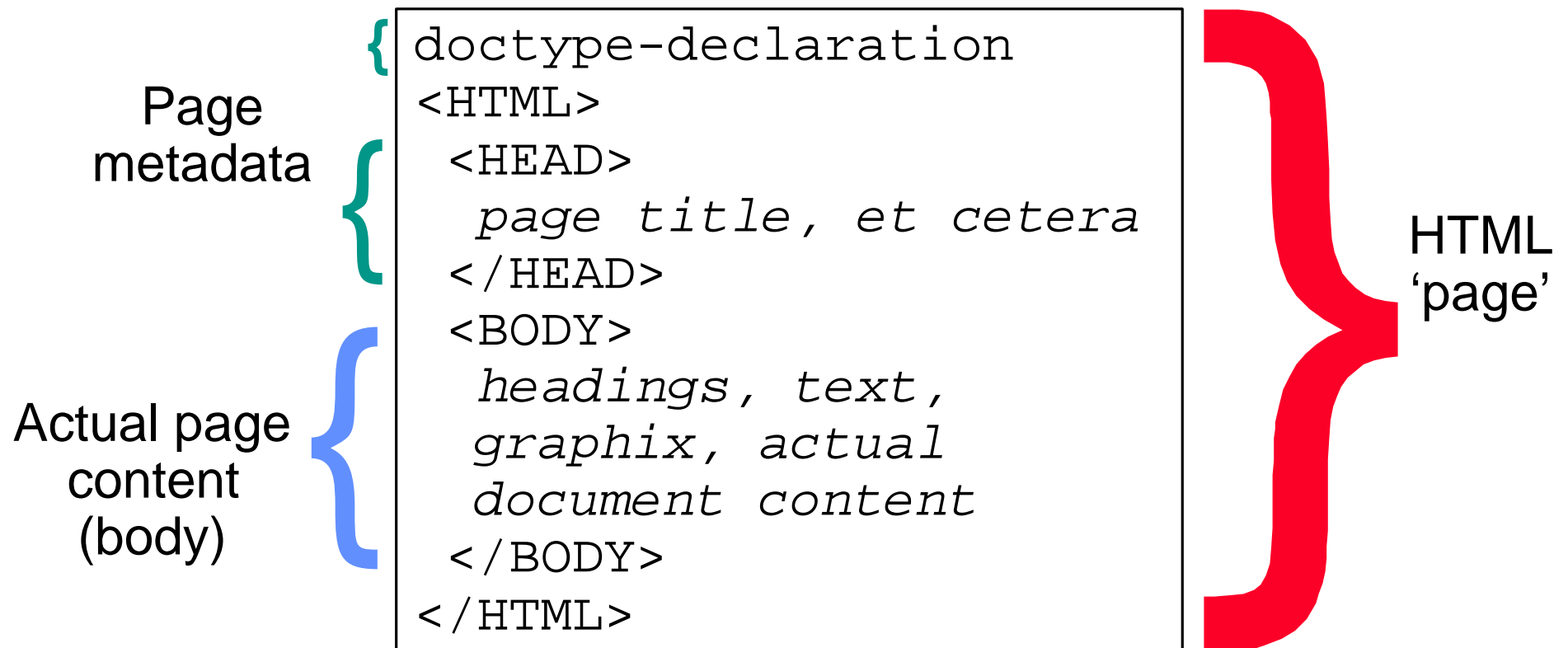
Client-Side Scripting

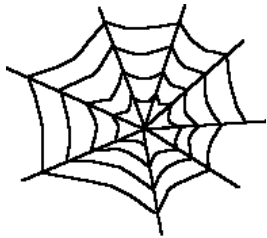
- JavaScript is probably the most widely used client-side scripting language
- Usually scripts are embedded in the HTML page itself
- Provisions exist for keeping scripts in separate files, but not all browsers support them
- JavaScript can provide run-time validation of form field contents
- ***Not all browsers support client-side scripting!***



The Well-Behaved HTML Page

or “If you’re going to emit HTML, emit *good* HTML”





Embedded JavaScript

- Appears in the HTML page in the `<HEAD></HEAD>` container
- Defined by `<SCRIPT></SCRIPT>` container
- Take care to keep unaware browsers from treating it as content:

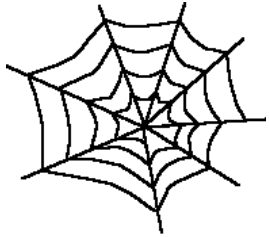
```
<SCRIPT LANGUAGE="JavaScript">
```

```
<!--
```

```
    [ statements ]
```

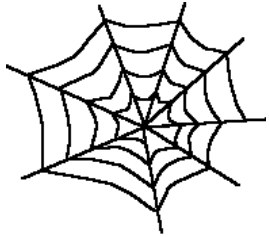
```
// -->
```

```
</SCRIPT>
```



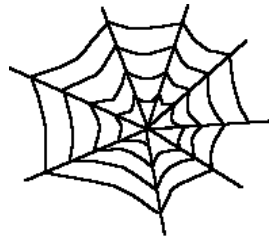
Java Applets

- Advantages
 - Capabilities are enormous
 - Everything a full language can do
- Disadvantages
 - Requires a client that can run Java!
 - Error reporting can be a problem
 - Requires an additional download from server
 - Java engine startup can be irritating



Testing Your Pages

- Before announcing the existence of your pages, you should check them out to verify that they look the way you want, and are accessible to your desired audience
- You should look at your pages with as many different browsers as you can (Netscape, Mosaic, Lynx, Cello, Internet Explorer, *et cetera*) - make sure the various appearances match your expectations, and that things are displayed intelligently
- Try to break things

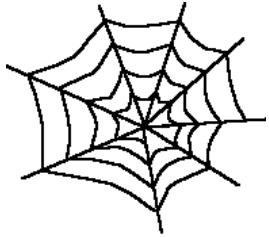


Testing Your Pages

(continued)

- Feed your HTML pages through the W3C HTML syntax validator at

`http://validator.w3.org/`



Going Further

- The Web Project pages:
<http://www.w3.org/>
- RFC2068, the HTTP/1.1 specification
- HTML Syntax Validator
<http://validator.w3.org/>
- CGI Specification:
<http://www.w3.org/pub/WWW/CGI/>
- CGI RFC project:
<http://Web.Golux.Com/coar/cgi/>
- USENET News

© 1998 by Ken A L Coar

MeepZor Consulting