

The `typehtml` package*

David Carlisle

1997/11/19

1 Introduction

This package enables the processing of HTML codes. The `\dohtml` command allows fragments of HTML to be placed within a \LaTeX document,

```
\dohtml
<html>
html markup ...
</html>
```

The `<html>...</html>` is *required*. (It is anyway a good idea to have these tags in an HTML document.)

The `\htmlinput` command is similar, but takes a file name as argument. In that case the file need not necessarily start and end with `<html>...</html>`.

This package covers most of the HTML2 DTD, together with the mathematics extensions from HTML3.¹ The rest of HTML3 may be added at a later date.

Its current incarnation has not been extensively tested, having been thrown together during the last couple of weeks in response to a question on `comp.text.tex` about the availability of such a package.

The package falls into three sections. Firstly the options section allows a certain amount of customisation, and enabling of extensions. Not all these options are fully operational at present. Secondly comes a section that implements a kind of SGML parser. This is not a real conforming SGML parser (not even a close approximation to such a thing!) The assumption (sadly false in the anarchic WWW) is that any document will have been validated by a conforming SGML parser before it ever gets to the stage of being printed by this package. Finally are a set of declarations that essentially map the declarations of the HTML DTD into \LaTeX constructs.

2 Options

2.1 HTML Level

The options `html2` (the default) and `html3` control HTML variant supported. Using the `html3` option will use up a lot more memory to support the extra

*This file has version number v0.12, last revised 1997/11/19.

¹The draft specification of HTML3 has expired, and the W3C group are currently devising a new proposed extension of HTML, so the mathematics typesetting part of this package may need substantial revision once a final specification of the HTML mathematics markup is agreed.

features, and the math entity (symbol) names. Against my better judgement there is also a `netscape` option to allow some of the non-HTML tags accepted by that browser.

2.2 Headings

The six options `chapter`, `chapter*`, `section`, `section*`, `subsection` and `subsection*` Determine to which L^AT_EX sectional command the HTML element `h1` is mapped. (`h2-h6` will automatically follow suit.) The default is `section*`.

2.3 Double Quote Handling

Most HTML pages use " as a quotation mark in text, for example:

```
quoted "like this" example
```

This slot in the ISO latin-1 encoding is for ‘straight’ double quotes. Unfortunately the Standard T_EX fonts in the OT1 encoding do not have such a character, only left and right quotes, “like this”. By default this package uses the `straightquotedbl` option which uses the L^AT_EX command `\textquotedbl` to render ". If used with the T1 encoded fonts `\usepackage[T1]{fontenc}` then the straight double quote from the current font is used. With OT1 fonts, the double quote is taken from the `\ttfamily` font, which looks "like this" which is fairly horrible, but better than the alternative which is ”like this”.

The `smartquotedbl` option redefines " so that it produces alternatively an open double quote “ then a close ”. As there is a chance of it becoming confused, it is reset to “ at the beginning of every paragraph, whatever the current mode.

Neither of these options affects the use of " as part of the SGML syntax to surround attribute values.

In principle the package ought to have similar options dealing with the single quote, but there the situation is more complicated due to its dual use as an apostrophe, so currently the package takes no special precautions: all single quotes are treated as a closing quote/apostrophe. Also the conventions of ‘open’ and ‘close’ quotes only really apply to English. If someone wants to suggest what the package should do with " in other languages...

2.4 Images

The default option is `imgalt` This means that all inline images (the HTML `img` element) are replaced by the text specified by the `alt` attribute, or `[image]` if no such attribute is specified.

The `imggif` option² uses the `\includegraphics` command so that inline images appear as such in the printed version.

The `imgps` option¹ is similar to `imggif` but first replaces the extension `.gif` at the end of the source file name by `.ps`. This will enable drivers that can not include GIF files to be used, as long as the user keeps the image in both PostScript and Gif formats.

²one day

2.5 Hyperref

Several options control how the HTML anchor tag is treated.

The default `nohyperref` option ignores name anchors, and typesets the body of src anchors using `\emph`.

The `ftnhyperref` option is similar to `nohyperref`, but adds a footnote showing the destination address of each link, as specified by the SRC attribute.

If the `hyperref` option is specified, the hypertext markup in the HTML file will be replicated using the hypertext specials of the HyperTeX group. If in addition the `hyperref` package is loaded, the extra features of that package may be used, for instance producing ‘native PDF’ specials for direct use by Adobe distiller rather than producing the specials of the hyperTeX conventions.

The `dviwindo` option converts the hypertext information in the HTML into the `\special` conventions of Y&Y’s *dviwindo* previewer for Microsoft Windows.

2.6 Big Integrals

L^AT_EX does not treat integral signs as variable sized symbols, in the way that it treats delimiters such as brackets. In common with summation signs and a few other operators, they come in just two fixed sizes, a small version for inline mathematics, and a large version used in displays. In fact by default L^AT_EX always uses the same two sizes (from the 10 pt math extension font) even if the document class has been specified with a size option such as `12pt`, or if a size command such as `\large` has been used.

The standard `exscale` package loads the math extension font at larger sizes if the current font size is larger than 10 pt.

The HTML3 math description explicitly states that integral signs should be treated like delimiters and stretch if applied to a large math expression. By default this package ignores this advice and treats integral signs in the standard way, however an option `bigint` does cause integral signs to ‘stretch’ (or at least be taken from a suitably large font). The standard Computer Modern fonts use a very ‘sloped’ integral which means that they are not really suitable for being stretched. Some other math fonts, for instance Lucida, have more vertical integral signs, and one could imagine in those cases making an integral sign with a ‘repeatable’ vertical middle section so that it could grow to an arbitrary size, in the way that brackets grow.

3 Latin-1 characters

The SGML character entities for the ISO-Latin1 characters such as `é` are recognised by this style, although as usual, some of them such as the Icelandic thorn character, `þ`, `\th`, produce an error if the old ‘OT1’ encoded fonts are being used. These characters will print correctly if ‘T1’ encoded fonts are used, for example by declaring `\usepackage[T1]{fontenc}`.

HTML also allows direct 8-bit input of characters according to the ISO-latin1 encoding, to enable this you need to enable latin-1 input for L^AT_EX with a declaration such as `\usepackage[latin1]{inputenc}`.

4 Mathematics

The HTML3 MATH element is fairly well supported, including the BOX and CLASS attributes. (Currently only CHEM value for class is supported, and as far as I can see the BOX attribute is only in the report, not in the dtd.) The super and subscripts are supported, including the shortref maps, however only the default right alignment is implemented so far. The convention described in the draft report for using white space to distinguish superscript positioning is fairly *horrible!*

The documentation that I could find on HTML3 did not include a full list of the entity names to be used for the symbols. This package currently *only* defines the following entities, which should be enough for testing purposes at least.

- `gt` (>) `lt` (<) (Already in the HTML2 DTD)
- Some Greek letters.
`alpha` (α) `beta` (β) `gamma` (γ) `Gamma` (Γ)
- Integral and Sum. `int` (\int) `sum` (Σ)
`int` (\int) `sum` (Σ)
- Braces (The delimiters `()`] also stretch as expected in the BOX element)
`lbrace` ($\{$) `rbrace` ($\}$)
- A random collection of mathematical symbols:
`times` (\times) `cup` (\cup) `cap` (\cap) `vee` (\vee) `wedge` (\wedge) `infty` (∞) `oplus` (\oplus) `ominus` (\ominus) `otimes` (\otimes)
- A Minimal set of trig functions:
`sin` (\sin) `cos` (\cos) `tan` (\tan)
- Also in the special context as attributes to ABOVE and BELOW elements the entities:
`overbrace` ($\overbrace{\hspace{1cm}}$) `underbrace` ($\underbrace{\hspace{1cm}}$) and any (T_EX) math accent name.

5 SGML Minimisation features

SGML (and hence HTML) support various minimisation features that aim to make it easier to enter the markup ‘by hand’. These features make the kind of ‘casual’ attempt at parsing SGML as implemented in this package somewhat error prone.

Two particular features are enabled in HTML. The so called SHORTTAG feature means that the name of a tag may be omitted if it may be inferred from the context. Typically in HTML this is used in examples like

```
<title>A Document Title</>
```

The end tag is shortened to `</>` and the system infers that TITLE is the element to be closed.

The second form of minimisation enabled in HTML is the OMITTAG feature. Here a tag may be omitted altogether in certain circumstances. A typical example

is the HTML list, where each list item is started with `` but the closing `` at the end of the item may be omitted and inferred by the following `` or `` tag.

This package is reasonably robust with respect to omitted tags. However it only makes a half hearted attempt at supporting the SHORTTAG feature. The TITLE example above would work, but nested elements, with multiple levels of minimised end tags will probably break this package.

It would be possible to build a \LaTeX system that had full knowledge of the HTML (or any other) DTD and in particular the ‘content model’ of every element, this would produce a more robust parsing system but would take longer than I was prepared to spend this week... In anycase if you need a fully conforming SGML parser, it probably makes sense to use an existing one (excellent free parsers are freely available) and then convert the output of the parser to a form suitable for \LaTeX . In that way all such concerns about SGML syntax features such as minimisation will have been resolved by the time \LaTeX sees the document.

6 Examples

6.1 A section

This document uses the `subsection*` option.

```
<h1>HTML and LaTeX</h1>
```

HTML and LaTeX

6.2 An itemised list

```
<ul>
<li> something
<li> something else
</ul>
```

- something
- something else

6.3 Latin1 Characters

```
&eacute; &ouml;
```

é ö

6.4 Images

Currently only the ALT attribute is supported.

```
This is an image of me 
```

This is an image of me DPC

6.5 A Form

```
<form
  action="http://www.cogs.susx.ac.uk/cgi-bin/ltxbugs2html"
  method=get><hr>
You can search for all the bug reports about: <select name="category">
<option>AMS LaTeX</option>
<option>Babel</option>
<option>Graphics and colour</option>
<option>LaTeX</option>
<option selected>Metafont fonts</option>
<option>PostScript fonts</option>
<option>Tools</option>
</select>
<hr>
</form>
```

You can search for all the bug reports about:

category
AMS LaTeX
Babel
Graphics and colour
LaTeX
• Metafont fonts
PostScript fonts
Tools

6.6 Styles of Mathematics

```
<math>
H_2_0 + CO_2_
</math>
<math class=chem>
H_2_0 + CO_2_
</math>
<math box>
H_2_0 + CO_2_
</math>
<math class=chem box>
H_2_0 + CO_2_
</math>
```

$H_2O + CO_2$

$H_2O + CO_2$

$H_2O + CO_2$

$H_2O + CO_2$

6.7 Integrals

Stretchy integrals with the `bigint` option.

```
<math>
{\&int;^1_3<left>
  1
  <over>
  {x+{1<over>x+{2<over>x+
    {3<over>x+{4<over>x}}}}}
<right>dx}
</math>
```

$$\int_3^1 \frac{1}{x + \frac{1}{x + \frac{2}{x + \frac{3}{x + \frac{4}{x}}}}} dx$$

And the same integral with the standard integral sign.

$$\int_3^1 \frac{1}{x + \frac{1}{x + \frac{2}{x + \frac{3}{x + \frac{4}{x}}}}} dx$$

6.8 Oversized delimiters

```
<math>
<box>(<left>1 <atop> 2 <right>)</box>
<box size=large>(<left>1 <atop> 2 <right>)</box>
</math>
```

$$\left(\begin{array}{c} 1 \\ 2 \end{array} \right) \left(\begin{array}{c} 1 \\ 2 \end{array} \right)$$

6.9 Roots, Overbraces etc

```
<math>
<above sym=overbrace> abc </above><sup>k</sup>
&emsp;
<root>3<of>x</root>
<sqrt>5</sqrt>
&emsp;
<below sym=underline> abc </below>
<above sym=widehat> abc </above>
</math>
```

$$\widehat{abc} \quad \sqrt[k]{x} \sqrt{5} \quad \underline{abc}$$

6.10 Arrays

```

<math>
aa<array align=top>
<row><item><text>first col</text><item><text>second col</text><item>
<text>third col</text><item><text>fourth col</text>
<row><item><text>row 2</text><item> a_22_ <item>a_23_<item>a_24_
<row><item><text>row 3</text><item rowspan=3 colspan=2>
a_32_-a_53_<item>a_34_
<row><item><text>row 4</text><item>a_44_
<row><item><text>row 5</text><item>a_54_
<row><item><text>row 6</text><item align=left>
al_62_<item>a_63_<item>a_64_
<row><item><text>row 7</text><item align=right>
ar_72_<item>a_73_<item>a_74_
</array>bb
</math>

```

<i>aa</i>	first col	second col	third col	fourth col <i>bb</i>
	row 2	a_{22}	a_{23}	a_{24}
	row 3	$a_{32} - a_{53}$		a_{34}
	row 4			a_{44}
	row 5			a_{54}
	row 6	al_{62}	a_{63}	a_{64}
	row 7	ar_{72}	a_{73}	a_{74}

Repeat that element, but change the ARRAY attributes as follows:

```

<array ldelim="(" rdelim=")" labels>

```

<i>aa</i>	first col	second col	third col	fourth col	<i>bb</i>
	row 2	a_{22}	a_{23}	a_{24})
	row 3	$a_{32} - a_{53}$		a_{34}	
	row 4			a_{44}	
	row 5			a_{54}	
	row 6	al_{62}	a_{63}	a_{64}	
	row 7	ar_{72}	a_{73}	a_{74}	

and finally an example of COLSPEC

```

<math>
<array colspec="R+C=L">
<row><item>abc_11_<item>abc_12_<item>abc_13_
<row><item>a_21_<item>a_22_<item>a_23_
<row><item>a_31_<item>a_32_<item>a_33_
</array>
</math>

```

$$\begin{array}{rcl}
 abc_{11} & + & abc_{12} = abc_{13} \\
 a_{21} & + & a_{22} = a_{23} \\
 a_{31} & + & a_{32} = a_{33}
 \end{array}$$

6.11 Tables

HTML3 tables are not yet supported, but there is a minimal amount to catch simple cases.

```
<table>
<caption>Simple Table</caption>
<tr><td>one <td> two
<tr><td>a <td> b
</table>
```

Simple Table

one	two
a	b

7 The Code

7.1 Option Handling

```
1 (*package)
2 \DeclareOption{html2}{\let\HTML@two@stop\endinput}

The # here, and in later option code will need doubling if you are using a LATEX
before June 95.
3 \DeclareOption{netscape}
4   {\def\HTML@not#1{\SGML@w{<#1> is not valid HTML}}}
5 \DeclareOption{html3}{\let\HTML@two@stop\relax}
6 \DeclareOption{nohyperref}{%
7   \let\HTML@doname\@secondoftwo
8   \def\HTML@dosrc#1#2{\emph{#2}}}
9 \DeclareOption{ftnhyperref}{%
10  \let\HTML@doname\@secondoftwo
11  \def\HTML@dosrc#1#2{\emph{#2}\footnote{HREF: \texttt{#1}}}}
12 \DeclareOption{hyperref}{%
13  \AtBeginDocument{%
14    \providecommand\href[2]{\special{html:<A href="#1">}}%
15    #2\special{html:</A>}}%
16    \providecommand\hypertarget[2]{\special{html:<A name="#1">}}%
17    #2\special{html:</A>}}%
18    \let\HTML@doname\hypertarget
19    \let\HTML@dosrc\href}}
```

Dviwindo itself deals with links within a document i.e., a src attribute of the form "#name". The code below detects a more general URL and fires a `launch:` action from the `\special`, which calls the non-existent command `typehtml`. Presumably this could be a batch file that calls netscape or some other WWW engine to process the URL.

```
20 \DeclareOption{dviwindo}{%
```

```

21 \def\HTML@dosrc#1#2{%
22   \leavevmode\sbbox\z@{#2}\count@ht\z@\@tempcnta\wd\z@
23   \if\string##\@car#1\@nil
24     \special{button: \the\@tempcnta\space\the\count@\space
25       "\@gobble#1"%
26   \else
27     \special{button: \the\@tempcnta\space\the\count@\space
28       launch: typehtml "#1"}
29   \fi
30   \special{color push}\special{color rgb 0 1 0}%
31   \unhbox\z@
32   \special{color pop}}}%
33 \def\HTML@doname#1#2{\leavevmode\special{mark: "#1"#2}}%
34 \DeclareOption{imgalt}{}
35 \DeclareOption{imggif}{\SGML@w{img gif support not done yet}}
36 \DeclareOption{imgps}{\SGML@w{img ps support not done yet}}
37 \DeclareOption{smartquotedbl}{%
38   \def\SGMLquotedbla{%
39     \textquotedblleft\global\let\SGMLquotedbl\SGMLquotedblb}
40   \def\SGMLquotedblb{%
41     \textquotedblright\global\let\SGMLquotedbl\SGMLquotedbla}
42   \let\SGMLquotedbl\SGMLquotedbla
43   \let\SGML@savedeverypar\everypar
44   \newtoks\everypar
45   \SGML@savedeverypar{%
46     \global\let\SGMLquotedbl\SGMLquotedbla\the\everypar}}
47 \DeclareOption{straightquotedbl}{%
48   \DeclareTextCommandDefault{\textquotedbl}{\ttfamily\char`\'}}%
49   \let\SGMLquotedbl\textquotedbl}
50 \DeclareOption{chapter}{%
51   \def\HTML@headings{%
52     \chapter\section\subsection%
53     \subsubsection\paragraph\subparagraph}}
54 \DeclareOption{chapter*}{%
55   \def\HTML@headings{%
56     {\chapter*}\section*}\subsection*}%
57     {\subsubsection*}\paragraph*}\subparagraph*}}
58 \DeclareOption{section}{%
59   \def\HTML@headings{%
60     \section\subsection%
61     \subsubsection\paragraph\subparagraph\endgraf}}
62 \DeclareOption{section*}{%
63   \def\HTML@headings{%
64     {\section*}\subsection*}%
65     {\subsubsection*}\paragraph*}\subparagraph*\endgraf}}
66 \DeclareOption{subsection}{%
67   \def\HTML@headings{%
68     \subsection%
69     \subsubsection\paragraph\subparagraph\endgraf\endgraf}}
70 \DeclareOption{subsection*}{%

```

```

71 \def\HTML@headings{%
72   {\subsection*}%
73   {\subsubsection*}{\paragraph*}{\subparagraph*}\endgraf\endgraf}}
74 \DeclareOption{bigint}{%
75   \let\HTML@int\int
76   \AtEndOfPackage{\RequirePackage{exscale}}}
77 \ExecuteOptions{section*,imgalt,html2,nohyperref,straightquotedbl}
78 \ProcessOptions

```

7.2 Fake SGML parser

```

79 \begingroup
80 \catcode'\<=\active
81 \catcode'\>=\active
82 \catcode'\&=\active
83 \catcode'\$=\active
84 \catcode'\"=\active
85 \catcode'\^=\active
86 \catcode'\_=\active
87 \catcode'\;=\active
88 \catcode'\A=\active
89 \catcode'\B=\active
90 \catcode'\C=\active
91 \catcode'\D=\active
92 \uccode'\A='\{%
93 \uccode'\B='\}%
94 \uccode'\C='|\%
95 \uccode'\D='\\%
96 \uppercase{\endgroup
\SGMLent@@
97 \def\SGMLent@@#1;{\csname SGML@E@#1\endcsname}
\SGMLent@@
98 \def\SGML@def@active#1>{%
99   \expandafter\def\csname SGML@#1\endcsname}
100 \def\dohtml{%
101   \begingroup
102   \ifx;\@undefined\expandafter\let\expandafter;\string;\fi
103   \ifx>\@undefined\expandafter\let\expandafter>\string>\fi
104   \catcode'\<=\active
105   \catcode'\>=\active
106   \catcode'\&=\active
107   \catcode'\{=\active
108   \catcode'\}=\active
109   \catcode'\$=\active
110   \catcode'\"=\active
111   \catcode'\^=\active
112   \catcode'\_=\active
113   \catcode'\|= \active
114   \catcode'\|= \active

```

```

115 \catcode\endlinechar=10
116 \catcode'\%=12
117 \catcode'\#=12
118 \catcode'\;=\active
119 \def\verbatim@nolig@list{\do\'\do\,\do\'\do\}-}
120 \def<\SgMLopen}%
121 \def&\SgMLent}%
122 \let^ \textasciicircum
123 \let~ \textasciitilde
124 \def_{\_}%
125 \let$ \$%
126 \def"\SgMLquotedbl}%
127 \def A{\}%
128 \def B{\}%
129 \def C{\texttt{|}}%
130 \def D{\texttt{\char'\}}%

```

Need to be careful about writing to table of contents.

```

131 \def\addcontentsline##1##2##3{%
132   {\def<\string<}\def&\string&}%
133   \addtocontents{##1}{\protect\dotochtml<html>}%
134   \addtocontents{##1}{\protect\contentsline{##2}{##3}{\thepage}}%
135   \addtocontents{##1}{\protect</html>}}%

```

`\dotochtml` A ‘compromise’ version of `\dohtml` for use in table of contents files. Allows HTML markup `<`, `&` etc, but also \TeX markup `\`, `{`, `}`. As these are incompatible, this is not 100% reliable but it seems to cover most cases in practice.

```

136 \def\dotochtml{%
137   \dohtml
138   \catcode'\z@
139   \catcode'\{@ne
140   \catcode'\}\tw@}

```

`\SgMLshortend`

```

141 \def\SgMLshortend{/}

```

`\SgMLgrab@`

```

142 \def\SgMLgrab@#1<#2>{%
143   \edef\@tempd{\lowercase{\def\noexpand\@tempd{\gobblespc#2 \relax}}}%
144   \@tempd
145   \ifx\@tempd\SgMLshortend\let\@tempd\@tempc\fi
146   \ifx\@tempb\@tempd
147     \advance\@tempcnta\@ne
148   \else
149     \ifx\@tempc\@tempd
150       \advance\@tempcnta\m@ne
151     \fi
152   \fi
153   \ifnum\@tempcnta=\z@
154     \expandafter\@tempa\expandafter{\the\@temptokena#1}%
155   \else
156     \addto@hook\@temptokena{#1<#2>}%
157     \expandafter\SgMLgrab@
158   \fi}

```

```

\SGMLopen
159 \def\SGMLopen#1>{%
160   \SGMLopen@#1 \@nil}
161 }

\htmlinput
162 \def\htmlinput#1{\dohtml\let\endhtml\relax\input{#1}\endgroup}

\gobblespc
163 \def\gobblespc#1 #2\relax{#1}

\SGMLgrabber
164 \def\SGMLgrabber#1#2{%
165   \def\@tempa{#2}%
166   \@tempcnta\@ne
167   \@temptokena{ }%
168   \lowercase{\def\@tempb{#1}\def\@tempc{/#1}}%
169   \SGMLgrab@}

\SGMLopen@
170 \begingroup
171 \catcode'\="= \active
172 \uppercase{\endgroup
173 \def\SGMLopen@#1 #2\@nil{%
174   \toks@{ }%
175   \edef\@tempa{\lowercase{\def\noexpand\SGMLElement{#1}}}\@tempa
176   \if!\@car#1\relax\@nil
177     \toks@{#1 #2}%
178     \SGML@w{Declaration ignored\MessageBreak<\the\toks@>\MessageBreak}%
179   \else
180     \if$#2$\else
181       \replacequotes#2"\@nil"%
182       \SGMLafterfi
183       \expandafter\toks@\expandafter{\expandafter}%
184       \expandafter\SGMLgetattrib\the\toks@ \@nil
185     \fi
186     \expandafter\ifx\csname SGML@\SGMLElement
187       \expandafter\endcsname\relax
188     \SGML@w{<\SGMLElement> undefined}%
189   \else
190     \csname SGML@\SGMLElement
191     \expandafter\expandafter\expandafter\endcsname
192   \fi
193 \fi}

\replacequotes
194 \def\replacequotes#1"#2"{%
195   \def\@tempb{#2}%
196   \ifx\@tempb\@nnil
197     \addto@hook\toks@{#1}%
198   \else
199     \addto@hook\toks@{#1{#2}}%
200   \expandafter\replacequotes
201   \fi}}

```

```

\SGMLafterfi
202 \def\SGMLafterfi#1\fi{\fi#1}

\SGMLgobbletofi
203 \def\SGMLgobbletofi#1\fi{\fi}

\SGMLgetattrib
204 \def\SGMLgetattrib#1 #2{%
205   \ifx\box#1\box\else
206     \SGMLgetval#1=$=\@nil
207     \def\@tempa{#2}%
208     \ifx\@tempa\@nnil
209       \expandafter\SGMLgobbletofi
210     \else
211       \expandafter\SGMLafterfi
212     \fi
213   \SGMLgetattrib#2%
214 \fi}

\SGMLgetval If no value was supplied #2 will be $ (Even if the value is $ The test is false, as that
              would be catcode 13. Done this way rather than looking for empty to distinguish
              alt="" with empty value.
215 \def\SGMLgetval#1=#2=#3\@nil{%
216   \ifcat$#2%
217     \lowercase{\SGML@addattrib\doimplied{#1}}%
218   \else
219     \lowercase{\SGML@addattrib{\do{#1}}}{#2}%
220   \fi}

\SGML@addattrib
221 \def\SGML@addattrib#1#2{\addto@hook\toks@{#1{#2}}}}

\SGML@w
222 \def\SGML@w{\PackageWarning{typehtml}}

\SGMLdef
223 \def\SGMLdef#1{%
224   \ifcat\noexpand#1\noexpand~%
225     \expandafter\SGML@def@active
226   \else
227     \expandafter\SGML@def
228   \fi}

\SGMLdef make sure this is a catcode 12 >.
229 \edef\@tempa{\def\noexpand\SGML@def##1\string>}\@tempa{%
230   \expandafter\def\csname SGML@#1\endcsname}

\SGMLent
231 \expandafter\def\expandafter\SGMLent\expandafter{%
232   \expandafter\protect\csname& \endcsname}

&
233 \expandafter\def\csname& \endcsname{%
234   \futurelet\@let@token\SGMLent@}

```

\SGMLent@

```
235 \def\SGMLent@{%
236 \ifx\@let@token\@sptoken
237 \&%
238 \else
239 \expandafter\SGMLent@@
240 \fi}
```

\SGMLentity

```
241 \def\SGMLentity#1{%
242 \expandafter\def\csname SGML@E@#1\endcsname}
```

7.3 The HTML2 DTD

```
243 \SGMLdef<html>{}
244 \SGMLdef</html>{\@endhtml}
245 \let\@endhtml\endgroup
246 \SGMLdef<title>{\typeout{***TITLE***}\SGMLgrabber{title}\typeout}
247 \long\def\@tempa#1#2#3#4#5#6{%
248 \SGMLdef<h1>{\SGMLgrabber{h1}{\HTMLsection{#1}}}%
249 \SGMLdef<h2>{\SGMLgrabber{h2}{\HTMLsection{#2}}}%
250 \SGMLdef<h3>{\SGMLgrabber{h3}{\HTMLsection{#3}}}%
251 \SGMLdef<h4>{\SGMLgrabber{h4}{\HTMLsection{#4}}}%
252 \SGMLdef<h5>{\SGMLgrabber{h5}{\HTMLsection{#5}}}%
253 \SGMLdef<h6>{\SGMLgrabber{h6}{\HTMLsection{#6}}}}
254 \expandafter\@tempa\HTML@headings
255 \def\HTMLsection#1#2{#1{\ignorespaces#2\unskip}}
256 \SGMLdef<head>{}
257 \SGMLdef</head>{}
258 \SGMLdef<body>{}
259 \SGMLdef</body>{}
260 \SGMLdef<bodytext>{}
261 \SGMLdef</bodytext>{}
262 \SGMLdef<p>{\par}
263 \SGMLdef</p>{\par}
264 \SGMLdef<blockquote>{\begin{quote}}
265 \SGMLdef</blockquote>{\end{quote}}
266 \SGMLdef<address>{\begin{quote}}
267 \SGMLdef</address>{\end{quote}}
268 \SGMLdef<ul>{\begin{itemize}}
269 \SGMLdef</ul>{\end{itemize}}
270 \SGMLdef<ol>{\begin{enumerate}}
271 \SGMLdef</ol>{\end{enumerate}}
272 \SGMLdef<li>{\item}
273 \SGMLdef</li>{}
274 \SGMLdef<dl>{}
275 \let\do\dldo
276 \let\doimplied\dlimplied
```

```

277 \begin{description}\the\toks@}
278 \SGMLdef</dl>{\end{description}}

279 \def\dldo#1#2{%
280 \def\@tempa{compact}\def\@tempb{#1}%
281 \ifx\@tempa\@tempb
282 \itemsep\z@
283 \advance\@totalleftmargin-\leftmargin
284 \advance\linewidth\leftmargin
285 \itemindent-\labelsep
286 \leftmargin\z@
287 \parshape \@ne \@totalleftmargin \linewidth
288 \fi}
289 \def\dlimplied#1{\dldo{#1}\relax}

290 \def\itx#1{\item[#1]}
291 \SGMLdef<dt>{\begin{lrbox}\z@\bfseries\let\maybeenddt\enddt}
292 \SGMLdef</dt>{\maybeenddt}
293 \SGMLdef<dd>{\maybeenddt}
294 \SGMLdef</dd>{}
295 \def\enddt{\end{lrbox}\item[\unhbox\z@]}
296 \let\maybeenddt\relax

297 \SGMLdef<a>{\SGMLgrabber{a}\HTML@anchor}

\HTML@anchor This handles the A tag.

298 \def\HTML@anchor#1{%
299 \let\@tempa@gobble
300 \def\_ {\string\_}%
301 \let\do\ado
302 \the\toks@
303 \@tempa{#1}}

\ado Thanks to SPQR for first pass at integrating hyperref.

304 \def\ado#1#2{%
305 \def\@tempb{name}\def\@tempc{#1}%
306 \ifx\@tempb\@tempc
307 \let\@tempa\@firstofone
308 \def\@tempa{\HTML@doname{#2}}%
309 \else
310 \def\@tempa{\HTML@dosrc{#2}}%
311 \fi}

312 \SGMLdef<pre>{%
313 \par
314 \begingroup
315 \parindent\z@
316 \obeylines\verbatim@font\@noligs
317 \frenchspacing\@vobeyspaces}
318 \SGMLdef</pre>{\endgroup}

319 \SGMLdef<tt>{\SGMLgrabber{tt}\texttt}
320 \SGMLdef<b>{\SGMLgrabber{b}\textbf}
321 \SGMLdef<i>{\SGMLgrabber{i}\textit}
322 \SGMLdef<em>{\SGMLgrabber{em}\emph}
323 \SGMLdef<strong>{\SGMLgrabber{strong}\textbf}

```



```

324 \SGMLdef<code>{\SGMLgrabber{code}\texttt}
325 \SGMLdef<samp>{\SGMLgrabber{samp}\textsf}
326 \SGMLdef<kbd>{\SGMLgrabber{kbd}\texttt}
327 \SGMLdef<var>{\SGMLgrabber{var}\textit}
328 \SGMLdef<cite>{\SGMLgrabber{cite}\textit}

329 \SGMLdef<form>{\par\medskip}
330 \SGMLdef</form>{\par\medskip}

331 \SGMLdef<select>{%
332   \let\do\selectdo
333   \the\toks@\par
334   \begin{tabular}{|l|}%
335     \hline@tempc\\\hline
336     \let\tabularnewline\relax
337     \ignorespaces}

338 \def\selectdo#1#2{%
339   \def\@tempa{name}\def\@tempb{#1}%
340   \ifx\@tempa\@tempb\def\@tempc{#2}\fi}

341 \SGMLdef</select>{\\\hline\end{tabular}}

342 \SGMLdef<option>{%
343   \gdef\optionbul{\phantom{$\bullet$}}%
344   \let\do\optiondo
345   \let\doimplied\optionimplied
346   \the\toks@
347   \tabularnewline
348   \let\tabularnewline\\%
349   \optionbul\space\ignorespaces}
350 \SGMLdef</option>{}

```

`\optiondo` Handle attributes to the OPTION element.

```

351 \def\optiondo#1#2{%
352   \def\@tempa{selected}\def\@tempb{#1}%
353   \ifx\@tempa\@tempb\gdef\optionbul{\bullet}\fi}

```

`\optionimplied` Handle the case where just the attribute value is given.

```

354 \def\optionimplied#1{%
355   \def\@tempa{selected}\def\@tempb{#1}%
356   \ifx\@tempa\@tempb\gdef\optionbul{\bullet}\fi}

```

```

357 \SGMLdef<input>{}

358 \SGMLdef<img>{%
359   \let\do\imgdo
360   \def\@tempa{\doimage}%
361   \the\toks@
362   \@tempa}}

363 \def\doimage{\textsf{[image]}}

```

`\imgdo` Handle IMG attributes (not very usefully)

```

364 \def\imgdo#1{\csname img=#1\endcsname}
365 \expandafter\def\csname img=align\endcsname#1{%
366   \SGML@w{align=#1 ignored}}
367 \expandafter\def\csname img=src\endcsname#1{%

```

```

368 \SGML@w{src=#1 ignored}}
369 \expandafter\def\csname img=height\endcsname#1{%
370 \SGML@w{height=#1 ignored}}
371 \expandafter\def\csname img=alt\endcsname#1{%
372 \def\doimage{#1}}

```

Horizontal rules and line breaks.

```

373 \SGMLdef<hr>{\par\smallskip\hrule\smallskip}
374 \SGMLdef<br>{\leavevmode\\}

```

These are obsolete in HTML3 but do them anyway.

```

375 \SGMLdef<xmp>{%
376 \SGML@pre
377 \def\@tempb{/xmp}%
378 \let\SGMLopen\HTML@xmptest}

379 \SGMLdef<listing>{%
380 \SGML@xmp
381 \def\@tempb{/listing}}

382 \SGMLdef<plaintext>{%
383 \SGML@xmp
384 \def\@tempb{/plaintext}}%

```

\HTML@xmptest

```

385 \def\HTML@xmptest#1>{%
386 \lowercase{\def\@tempa{#1}}%
387 \ifx\@tempa\@tempb
388 \endgroup
389 \else
390 \SGMLafterfi
391 <#1>%
392 \fi}

```

SGML syntax Character entities.

```

393 \SGMLentity{amp}{\&}
394 \SGMLentity{lt}{\ensuremath{<}}
395 \SGMLentity{gt}{\ensuremath{>}}

```

ISO Latin-1 Character entities.

```

396 \SGMLentity{aacute}{\'a}
397 \SGMLentity{Aacute}{\'A}
398 \SGMLentity{acirc}{\^a}
399 \SGMLentity{Acirc}{\^A}
400 \SGMLentity{agrave}{\`a}
401 \SGMLentity{Agrave}{\`A}
402 \SGMLentity{aring}{\r a}
403 \SGMLentity{Aring}{\r A}
404 \SGMLentity{atilde}{\~a}
405 \SGMLentity{Atilde}{\~A}
406 \SGMLentity{auml}{\`a}
407 \SGMLentity{Auml}{\`A}
408 \SGMLentity{aelig}{\ae}
409 \SGMLentity{Aelig}{\AE}
410 \SGMLentity{ccedil}{\c c}
411 \SGMLentity{Ccedil}{\c C}

```

```

412 \SGMLentity{eth}{\dh}
413 \SGMLentity{ETH}{\DH}
414 \SGMLentity{eacute}{\'e}
415 \SGMLentity{Eacute}{\'E}
416 \SGMLentity{ecirc}{\`e}
417 \SGMLentity{Ecirc}{\`E}
418 \SGMLentity{egrave}{\`e}
419 \SGMLentity{Egrave}{\`E}
420 \SGMLentity{euuml}{\`e}
421 \SGMLentity{Euuml}{\`E}
422 \SGMLentity{iacute}{\`i}
423 \SGMLentity{Iacute}{\`I}
424 \SGMLentity{icirc}{\`i}
425 \SGMLentity{Icirc}{\`I}
426 \SGMLentity{igrave}{\`i}
427 \SGMLentity{Igrave}{\`I}
428 \SGMLentity{iuml}{\`i}
429 \SGMLentity{Iuml}{\`I}
430 \SGMLentity{ntilde}{\~n}
431 \SGMLentity{Ntilde}{\~N}
432 \SGMLentity{oacute}{\`o}
433 \SGMLentity{Oacute}{\`O}
434 \SGMLentity{ocirc}{\`o}
435 \SGMLentity{Ocirc}{\`O}
436 \SGMLentity{ograve}{\`o}
437 \SGMLentity{Ograve}{\`O}
438 \SGMLentity{oslash}{\oe}
439 \SGMLentity{Oslash}{\OE}
440 \SGMLentity{otilde}{\~o}
441 \SGMLentity{Otilde}{\~O}
442 \SGMLentity{ouml}{\`o}
443 \SGMLentity{Ouml}{\`O}
444 \SGMLentity{szlig}{\ss}
445 \SGMLentity{thorn}{\th}
446 \SGMLentity{THORN}{\TH}
447 \SGMLentity{uacute}{\`u}
448 \SGMLentity{Uacute}{\`U}
449 \SGMLentity{ucirc}{\`u}
450 \SGMLentity{Ucirc}{\`U}
451 \SGMLentity{ugrave}{\`u}
452 \SGMLentity{Ugrave}{\`U}
453 \SGMLentity{uuml}{\`u}
454 \SGMLentity{Uuml}{\`U}
455 \SGMLentity{yacute}{\`y}
456 \SGMLentity{Yacute}{\`Y}
457 \SGMLentity{yuml}{\`y}

```

7.4 Netscape Non-HTML tags

Netscape allows certain tags that do not correspond to HTML elements. These are *Bad Thing*. Originally the documentation of this package stated that such ‘extensions’ would not be supported, however as a request came from³ who

³Name withheld to protect the guilty

also supplied most of the code in this section (and also the table section), I have added some support which is enabled if the `netscape` option is used.

```
458 \ifx\HTML@not@\undefined\else
```

Do something with bad reprehensible nonstandard tags that have the annoying habit of turning up often in html files that I want to print. [mjd,1996/03/20]

`\HTML@not` is defined above in the `netscape` option: Naughty Nonstandard Extension Warning for things like `<center>` and ``. (I thought these were Netscape-specific but the technical notes at Spyglass's web site showed that I was wrong. [mjd,1996/03/20])

```
459 \SGMLdef<center>{\HTML@not{center}\begin{center}}
```

```
460 \SGMLdef</center>{\end{center}}
```

```
461 \SGMLdef<blink>{\SGMLgrabber{blink}\textbf}
```

```
462 \SGMLdef<font>{\HTML@not{font}\begin{group}}
```

```
463 \let\do\fontdo\the\toks@}
```

```
464 \SGMLdef</font>{\end{group}}
```

`\fontdo` must look at the first character of the 'size, value to see if it is a relative size change (+ or -). Otherwise it is an absolute size change.

```
465 \def\fontdo#1#2{%
```

```
466 \def\@tempa{size}\def\@tempb{#1}%
```

```
467 \ifx\@tempa\@tempb
```

```
468 \font@switch#2\relax\@nil
```

```
469 \fi}
```

Let's hack a nice little hook into `\@setfontsize` (tsk tsk). If we can set the current font size number there, it makes the rest of the job much easier.

```
470 \toks@\expandafter{\set@fontsize{#1}{#2}{#3}}
```

```
471 \edef\@tempa{%
```

```
472 \def\noexpand\set@fontsize##1##2##3{\the\toks@\noexpand\set@fontnum}}
```

```
473 \@tempa
```

Take `\f@size` which is a real number, convert it to an integer, and normalize to the desired range.

```
474 \def\set@fontnum{\dimen@\f@size\p@
```

```
475 \dimen@\mul@ptsize\dimen@
```

```
476 \count@\dimen@ \divide\count@\p@
```

```
477 \advance\count@ -5\relax
```

```
478 \edef\@fontnum{\number\count@}}
```

Nice consistent naming conventions as always. multiplier if 11pt or 12pt documentclass option is used

```
479 \def\mul@ptsize{}
```

```
5 = \normalsize I think
```

```
480 \def\@fontnum{5}
```

```
Initialize \mul@ptsize
```

```
481 \ifcase 0\@ptsize\relax
```

```
482 \global\let\mul@ptsize\@empty% case 0, ptsize = 10
```

```
483 \or\gdef\mul@ptsize{.9091}% case 1, ptsize = 11
```

```
484 \else\gdef\mul@ptsize{.8333}% case 2, ptsize = 12
```

```
485 \fi
```

`\font@switch` looks for + or - and selects a suitable fontsize command.

```

486 \def\font@switch#1#2\@nil{\count@\fontnum\relax
487 \ifx +#1\advance\else\ifx -#1\advance\fi\fi
488 \count#1#2\relax
489 \ifcase\count@ \tiny\or \tiny\or \scriptsize
490 \or\footnotesize \or\small \or\normalsize \or\large
491 \or\Large \or\LARGE \or\huge \else\Huge \fi}
492 \fi

```

7.5 The HTML3 DTD

`\HTML@two@stop` is `\endinput` (and so the package stops here) unless the HTML3 option is given.

```

493 \HTML@two@stop
494 \SGML@w{HTML3 support not finished yet}
495 \SGMLdef<math>{\SGMLgrabber{math}\domath}
496 \SGMLdef<sup>{\^ \bgroup\HTMLscriptmap}
497 \SGMLdef</sup>{\egroup}
498 \SGMLdef<sub>{\_ \bgroup\HTMLscriptmap}
499 \SGMLdef</sub>{\egroup}

```

GRUMBLE! GRUMBLE! GRUMBLE! Possibly the worst feature of T_EX's math markup is the nature of the infix operators for fractions and the like. And here it is faithfully (or actually not very faithfully) reconstructed here...

```

500 \SGMLdef<box>{\SGMLgrabber{box}\dobox}
501 \begingroup
502 \catcode'\<=\active
503 \catcode'\>=\active
504 \catcode'\&=\active
505 \catcode'\_=\active
506 \catcode'\^=\active
507 \catcode'\"=\active

```

`\domath` Handle the MATH element. The body is pre-expanded one level to replace { } by BOX elements, and to replace any SGML entity references by single T_EX tokens so they can be recognised more easily. Then start math mode with `\[` (which may have been redefined locally if the BOX attribute was used) set up the shorteref map.

```

508 \gdef\domath#1{%
509  {{\def&{\expandafter\expandafter\expandafter\noexpand\SGMLent@}}%
510  \let<\relax\let>\relax\let_\relax\let^\relax\let"\relax
511  \def\{<box>\def\}{</box>}%
512  \xdef\@gtempa{#1}%
513  \let\do\mathdo
514  \let\doimplied\mathimplied
515  \the\toks@
516  \[%
517  \m@th\nulldelimiterspace\z@
518  \def^{<sup>}%
519  \def_{<sub>}%
520  \@gtempa\}}

```

```

\HTMLscriptmap Set up the shortref map used in super and subscripts.
521 \gdef\HTMLscriptmap{%
522   \def^{\</sup>}%
523   \def_{\</sub>}}

\doebox Handle the BOX element. First deal with the attributes, then set up the shortref
map. Then start looking for a LEFT tag.
524 \gdef\doebox#1{%
525   {\let\do\boxdo
526    \let\bigstrut\relax
527    \the\toks@
528    \def^{\<sup>}%
529    \def_{\<sub>}%
530    \lookleft@#1<left>\@nil}}

\lookleft@ See whether this BOX element contains a LEFT tag. Supply a ‘null delimiter’ if
not one supplied.
531 \gdef\lookleft@#1<left>#2\@nil{%
532   \if$#2$%
533     {\left.\bgroup#1\mayberight}%
534   \else
535     \lookbox@#1<box>\@nil#2\@nil
536   \fi}

\lookbox@ Having found a LEFT tag, need to check it isn’t inside a nested BOX. The following
code looks for an explicit <BOX> (which includes a { shortref as that will have been
expanded by now, however it will fail if nested boxes have attributes, so it may
need some further modifications later.
537 \gdef\lookbox@#1<box>#2\@nil#3<left>\@nil{%
538   \if$#2$%
539     {\maybeleft#1\@nil#3\mayberight}
540   \else
541     {#1 \boxtofront#2 <left> #3}%
542   \fi}

\boxtofront After all that messing around need to put the BOX tag back where we found it.
543 \gdef\boxtofront#1<box>{\<box>#1}

544 \endgroup

\mathdo
545 \def\mathdo#1#2{%
546   \def\@tempa{class-chem}\def\@tempb{#1-#2}%
547   \ifx\@tempa\@tempb
548     \everymath{\fam\z@\}\everydisplay{\fam\z@\}%
549   \fi}
550 \def\mathimplied#1{%
551   \def\@tempa{box}\def\@tempb{#1}%
552   \ifx\@tempa\@tempb
553     \def\[\{\center\setbox\z@\hbox\bgroup$\displaystyle}%
554     \def\][{\egroup\fbbox{\box\z@}\endcenter}%
555   \fi}

```

```

\boxdo
556 \def\boxdo#1#2{%
557   \def\@tempa{size}\def\@tempb{#1}%
558   \ifx\@tempa\@tempb
559     \def\@tempb{#2}
560     \def\@tempa{normal}\ifx\@tempa\@tempb\def\@tempc{1}\fi
561     \def\@tempa{medium}\ifx\@tempa\@tempb\def\@tempc{2}\fi
562     \def\@tempa{large}\ifx\@tempa\@tempb\def\@tempc{3}\fi
563     \def\@tempa{huge}\ifx\@tempa\@tempb\def\@tempc{4}\fi
564     \edef\bigstrut{\vrule\@height\@tempc\ht\strutbox\@width\z@}
565   \fi}

\SGML@left
566 \SGMLdef<left>{\left.\bgroup}

\maybeleft
567 \def\maybeleft{\egroup\bigstrut\right.}

\maybeleft
568 \def\maybeleft#1#2\@nil{%
569   \in@{#1}{( ) [] \SGML@E@rbrace \SGML@E@lbrace}%
570   \ifin@
571     \left#1\bgroup#2%
572   \else
573     \let\SGML@E@int\HTML@bigint
574     #1#2\left.\bgroup\let\SGML@E@int\int
575   \fi}

\rightttest
576 \def\rightttest#1{%
577   \in@{#1}{( ) [] \SGML@E@rbrace \SGML@E@lbrace}%
578   \ifin@
579     \right#1\let\maybeleft\relax
580   \else
581     \right.\let\maybeleft\relax\expandafter#1%
582   \fi}

```

7.6 ‘Big int’ processing

I am not sure that stretchy integral signs are good idea in general, and certainly they do not fit well with the Computer Modern style of sloping integral sign as opposed to the more vertical style of, say, Lucida. However...

```

\HTML@int
583 \ifx\HTML@int\@undefined

\HTML@bigint Normally just use the standard \int.
584 \let\HTML@bigint\int
585 \else

```

With the `bigint` option . The original `\int` (in a big font) together with any saved limits (in the normal font).

```
586 \def\HTML@int{\int^{\box\tw@}_{\box4}}
```

\HTML@bigint

```
587 \def\HTML@bigint#1\left.\bgroup{%
588   \def\@tempa{#1}%
589   \setbox\z@\hbox\bgroup
590     \aftergroup\HTMLafterbigint$\displaystyle\bgroup
591     \aftergroup$\aftergroup\egroup}
```

\HTMLafterbigint

```
592 \def\HTMLafterbigint{%
593   \dimen@.5\ht\z@
594   \advance\dimen@.5\dp\z@
595   {\SGMLdef<sup>{\setbox\tw@\hbox\bgroup\HTMLscriptmap$\scriptstyle}%
596     \SGMLdef<sub>{\setbox4\hbox\bgroup\HTMLscriptmap$\scriptstyle}%
597     \SGMLdef</sup>{\egroup}%
598     \SGMLdef</sub>{\egroup}%
599     \setbox\tw@\box\voidb@x
600     \setbox4\box\voidb@x
601     \@tempa
602     \ifdim\dimen@>\f@size\p@
```

At this point, could do `\fontsize\dimen@z@\selectfont` but that would load *all* the math fonts at a strange size, so instead just load the extension font, and then subvert NFSS to drop that into the math expression. The NFSS interface is still used to declare the font so that a size substitution is done on the loading (otherwise every integral may use up a new font).

```
603     \mathop{\hbox{\DeclareFixedFont\@tempa{OMX}{cmex}{m}{n}\dimen@
604       $\displaystyle\textfont\thr@@\@tempa\HTML@int$}}%
605   \else
606     \HTML@int
607   \fi
608 } \left.\box\z@}
609 \fi
```

See above grumble. The HTML3 DTD comments specifically refer to these as ‘`LATEX` commands’ but they are no such thing. They are in plain and survive into `LATEX` under protest! The AMS `LATEX` documentation contains a much longer diatribe against these infix commands, and they are *disabled* in the AMS `LATEX` styles.

```
610 \SGMLdef<over>{\over}
611 \SGMLdef<atop>{\atop}
612 \SGMLdef<choose>{\choose}
613 \SGMLdef<right>{\egroup\bigstrut\rightttest}
614 \SGMLdef<above>{\SGMLgrabber{above}%
615   {\let\@tempc\overlineop
616     \let\do\abovedo
617     \the\toks@
618     \@tempc}}
619 \SGMLdef<below>{\SGMLgrabber{below}%
620   {\let\@tempc\underlineop
621     \let\do\abovedo
622     \the\toks@
623     \@tempc}}
```



```

624 \def\overlineop#1{\mathop{\overline{#1}}}
625 \def\underlineop#1{\mathop{\underline{#1}}}

626 \def\abovedo#1#2{%
627   \def\@tempa{sym}\def\@tempb{#1}%
628   \ifx\@tempa\@tempb\def\@tempc{\csname#2\endcsname}\fi}

629 \SGMLdef<vec>{\SGMLgrabber{vec}\vec}
630 \SGMLdef<bar>{\SGMLgrabber{bar}\bar}
631 \SGMLdef<dot>{\SGMLgrabber{dot}\dot}
632 \SGMLdef<ddot>{\SGMLgrabber{ddot}\ddot}
633 \SGMLdef<hat>{\SGMLgrabber{hat}\hat}
634 \SGMLdef<tilde>{\SGMLgrabber{tilde}\tilde}

635 \SGMLdef<t>{\SGMLgrabber{t}\mathrm}
636 \SGMLdef<bt>{\SGMLgrabber{bt}\mathbf}

637 \SGMLdef<text>{\SGMLgrabber{text}\textnormal}%% not in the dtd????

638 \SGMLdef<root>{\rootfudge}
639 \def\rootfudge#1{%
640   \setbox\rootbox\hbox\bgroup$\m@th\scriptscriptstyle\bgroup#1}

    I think the HTML3 DTD is wrong here4, it allows the OF element to take
    content, which is at variance with the description in the text.

641 \SGMLdef<of>{\egroup$\egroup\SGMLgrabber{root}\offfudge}
642 \SGMLdef</of>{ }
643 \def\offfudge#1{\mathpalette\r@@t{#1}}
644 \SGMLdef<sqrt>{\SGMLgrabber{sqrt}\sqrt}

    Hate allocating registers, so this will probably go, but for now give myself four
    (global) count registers to play with.

645 \newcount\HTMLrow
646 \newcount\HTMLcol
647 \newcount\HTMLrowspan
648 \newcount\HTMLcolspan

    The HTML array element. Support for ALIGN, COLSPAN, ROWSPAN LA-
    BELS, LDELIM and RDELIM. However not all combinations of alignment and
    labels do ‘the right thing’.

    Uses a TEX primitive \halign construction, rather than use the LATEX array
    environment directly.

649 \SGMLdef<array>{\ifnum' }=0\fi
650   \let\do\arraydo
651   \let\doimplied\arrayimplied
652   \let\HTMLal.%
653   \let\HTMLar.%
654   \global\HTMLrow\z@
655   \let\HTMLabox\vcenter
656   \the\toks@
657   \setbox\z@\vbox\bgroup\halign\bgroup
658     \strut\span\HTMLacolspec\cr\nocr}

```

\HTMLacolspec

```
659 \def\HTMLacolspec{##&&##}
```

⁴Since confirmed by Dave Raggett, the HTML3 author

\HTMLamakepream

```
660 \def\HTMLamakepream#1{%
661   \let\HTMLacolspec\@empty
662   \let\@sharp\relax
663   \lowercase{\@tfor\@tempc:=#1}\do{%
664     \if\@tempc l%
665       \edef\HTMLacolspec{\HTMLacolspec\@sharp\hfill&}%
666     \else
667       \if\@tempc c%
668         \edef\HTMLacolspec{\HTMLacolspec\hfill\@sharp\hfill&}%
669       \else
670         \if\@tempc r%
671           \edef\HTMLacolspec{\HTMLacolspec\hfill\@sharp&}%
672         \else
673           \if\@tempc +%
674             \edef\HTMLacolspec{\HTMLacolspec$+$}%
675           \else
676             \if\@tempc -%
677               \edef\HTMLacolspec{\HTMLacolspec$-$}%
678             \else
679               \if\@tempc =%
680                 \edef\HTMLacolspec{\HTMLacolspec$=$}%
681               \fi
682             \fi
683           \fi
684         \fi
685       \fi
686     \fi}%
687 \def\@sharp{#####}%
688 \edef\HTMLacolspec{\HTMLacolspec&\@sharp}}

689 \SGMLdef</array>{\HTMLendarray}
690 \let\HTMLcr\cr
```

\HTMLendarray Non LABELS ending

```
691 \def\HTMLendarray{%
692   \endi\crr\egroup\egroup
693   \ifx\HTMLabox\vtop
694     \setbox\z@\vtop{\unvbox\z@}%
695   \else
696     \ifx\HTMLabox\vcenter
697       \dimen@ \ht\z@
698       \advance\dimen@\dp\z@
699       \divide\dimen@\tw@
700       \advance\dimen@-\ht\z@
701       \setbox\z@\hbox{\raise\dimen@\box\z@}%
702     \fi
703   \fi
704   \dimen@=\ht\z@
705   \setbox\z@
706   \hbox{$\left\HTMLal\kern-1em\right\HTMLar$}%
707   \advance\dimen@-\ht\z@
708   \raise\dimen@\box\z@
709   \ifnum'={0\fi}}
```

`\HTMLendarraylabels` LABELS ending

```
710 \def\HTMLendarraylabels{%
711   \endi\crr\strut\cr\egroup\egroup
712   \setbox2=\vsplit\z@ to \baselineskip
713   \setbox\z@\vbox{\unvbox\z@\global\setbox\@ne\lastbox}%
714   \setbox4\hbox{\unhbox\@ne\unskip\global\setbox\@ne\lastbox}%
715   \vcenter{%
716     \box2
717     \hbox{${\kern\wd\@ne
718       \left\HTMLal\kern-\wd\@ne
719         \vcenter{\box\z@}}%
720     \right\HTMLar$}}%
721   \ifnum'={0\fi}}

722 \def\nocr{\relax\iffalse{\fi\let\HTMLcr\relax\iffalse}\fi}

723 \SGMLdef<row>{%
724   \endi\HTMLcr
725   \global\advance\HTMLrow\@ne
726   \global\HTMLcol\z@}

727 \SGMLdef</row>{}

728 \SGMLdef<item>{%
729   \let\do\itemdo
730   \gdef\@gtempa{\global\advance\HTMLcol\@ne}%
731   \gdef\@gtempb{}%
732   \gdef\@gtempc{}%
733   \global\HTMLcolspan\@ne
734   \the\toks@
735   \endi%
736   \@gtempc
737   \@gtempa
```

If an earlier row contained an entry spanning down to this point, need to jump across to the next column (and perhaps further).

```
738 \spanifneeded
```

`\@gtempb` is normally empty but will be defined if the item had an ALIGN attribute.

```
739 \@gtempb
```

First box each entry which allows measuring needed (but not yet done) for vertical spanning.

```
740 \setbox\z@\hbox\bgroup$%
741 \def\endi{\unskip$\egroup%
742   \quad\HTMLalleft\box\z@\HTMLaright\quad&}%
743 \ignorespaces}
```

`\spanifneeded` If the current row/column is in the list of spanned entries, jump to next column and look again.

```
744 \def\spanifneeded{%
745   \edef\@tempa{\noexpand\in@{\the\HTMLrow/\the\HTMLcol,}\spanititems}}%
746   \@tempa
747   \ifin@
748     \@firstofone{&}\global\advance\HTMLcol\@ne
```

```

749 \expandafter\spanifneeded
750 \fi}

```

As usual handle end tags that may be omitted by making them translate to empty.

```
751 \SGMLdef</item>{}
```

`\HTMLaleft` Default stuff to put around the entries. Locally redefined by an ALIGN attribute.

```

\HTMLaright 752 \let\HTMLaleft\hfil
753 \let\HTMLaright\hfil

```

`\endi` Code to end an item. Extra indirection used to handle omitted tags.

```
754 \let\endi\relax
```

`\arraydo` Handle ARRAY attributes.

```

755 \def\arraydo#1#2{%
756   \def\@tempa{#1}\def\@tempb{#2}%
757   \def\@tempc{align}%
758   \ifx\@tempa\@tempc
759     \def\@tempc{top}%
760     \ifx\@tempb\@tempc
761       \let\HTMLabov\vtop
762     \else
763       \def\@tempc{bottom}%
764       \ifx\@tempb\@tempc
765         \let\HTMLabov\vbox
766       \fi
767     \fi
768   \else
769     \def\@tempc{ldelim}%
770     \ifx\@tempa\@tempc
771       \let\HTMLal\@tempb
772     \else
773       \def\@tempc{rdelim}%
774       \ifx\@tempa\@tempc
775         \let\HTMLar\@tempb
776       \else
777         \def\@tempc{labels}%
778         \ifx\@tempa\@tempc
779           \let\HTMLendarray\HTMLendarraylabels
780         \else
781           \def\@tempc{colspec}%
782           \ifx\@tempa\@tempc
783             \HTMLamakepream{#2}%
784           \fi
785         \fi
786       \fi
787     \fi
788   \fi}

```

`\arrayimplied`

`\itemdo` Handle ITEM attributes

```

789 \def\itemdo#1#2{%
790   \def\@tempa{#1}\def\@tempb{#2}%
791   \def\@tempc{colspan}%
792   \ifx\@tempa\@tempc
793     \global\HTMLcolspan#2\relax
794     \gdef\@gtempa{\@multispan#2\relax\global\advance\HTMLcol#2\relax}%
795   \else
796     \def\@tempc{align}%
797     \ifx\@tempa\@tempc
798       \def\@tempc{left}%
799       \ifx\@tempb\@tempc
800         \gdef\@gtempb{\let\HTMLaleft\relax}%
801       \else
802         \def\@tempc{right}%
803         \ifx\@tempb\@tempc
804           \gdef\@gtempb{\let\HTMLaright\relax}%
805         \fi
806       \fi
807     \else
808       \def\@tempc{rowspan}%
809       \ifx\@tempa\@tempc
810         \global\HTMLrowspan#2\relax
811         \gdef\@gtempc{%
812           \@tempcnta=\HTMLrow
813           \advance\@tempcnta\HTMLrowspan

```

Double loop adds all the entries below this a ROWSPAN entry to `\spanitems` list.

```

814     \loop
815       \@tempcntb=\HTMLcol
816       \advance\@tempcntb\HTMLcolspan
817       \advance\@tempcnta\m@ne
818       \ifnum\@tempcnta>\HTMLrow
819         {\loop
820           \xdef\spanitems{%
821             \spanitems\the\@tempcnta/\the\@tempcntb,}%
822           \advance\@tempcntb\m@ne
823           \ifnum\@tempcntb>\HTMLcol
824             \repeat}%
825         \repeat}%
826     \fi
827   \fi
828 \fi}

```

`\spanitems` Initial value for list of spanned entries.

```

829 \def\spanitems{,}

830 \SGMLentity{thinsp}{\,}
831 \SGMLentity{emsp}{\quad}

```

Far from final list of math symbol entity names...

```

832 \SGMLentity{alpha}{\alpha}
833 \SGMLentity{beta}{\beta}
834 \SGMLentity{gamma}{\gamma}
835 \SGMLentity{Gamma}{\Gamma}

```

```

836 \SGMLentity{int}{\int}
837 \SGMLentity{sum}{\sum}

838 \SGMLentity{lbrace}{\lbrace}
839 \SGMLentity{rbrace}{\rbrace}

840 \SGMLentity{times}{\times}
841 \SGMLentity{cup}{\cup}
842 \SGMLentity{cap}{\cap}
843 \SGMLentity{vee}{\vee}
844 \SGMLentity{wedge}{\wedge}
845 \SGMLentity{infty}{\infty}
846 \SGMLentity{oplus}{\oplus}
847 \SGMLentity{ominus}{\ominus}
848 \SGMLentity{otimes}{\otimes}

849 \SGMLentity{sin}{\sin}
850 \SGMLentity{cos}{\cos}
851 \SGMLentity{tan}{\tan}

```

8 HTML3 Tables

Not done yet, but here is a start...

Final version will probably need primitive `\halign` coding as for (but hopefully better than) array stuff above. Also will need to be lontable-like.

This is all very slapdash and temporary [mjd,1996/03/20]. Don't expect good-looking results, just results, occasionally.

```

852 \SGMLdef<table>{\begin{table}[htp]\centering\begin{tabular}{*{10}c}}
853 \SGMLdef</table>{\end{tabular}\end{table}}
854 \SGMLdef<tr>{\ifhmode\expandafter\\fi\relax}
855 \SGMLdef</tr>{\\relax}
856 \SGMLdef<td>{\ifvmode\else\expandafter\hiddenamp\fi}
857 \def\hiddenamp{&}

```

if `<td>` is present for each cell, then `</td>` doesn't need to do anything

```
858 \SGMLdef</td>{}
```

Whoa, if I'm to define caption properly I'd have to look up how/where it's used. Who, lazy old me?

```

859 \SGMLdef<caption>{\end{tabular}\begingroup\bfseries}
860 \SGMLdef</caption>{\endgroup\par\smallskip\begin{tabular}{*{10}{c}}}}
861 </package>

```