# The **typehtml** package[*]

David Carlisle

1997/11/19

## 1 Intoduction

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.

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

[1] The 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.

# 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 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 LaTeX 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 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 TeX 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 LaTeX 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` option[2] uses the `\includegraphics` command so that inline images appear as such in the printed version.

The `imgps` option[1] 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.

## 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

LATEX 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 LATEX 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.

[2] one day

# 3 Latin-1 characters

The SGML character entities for the ISO-Latin1 characters such as `&eacute;` are recognised by this style, although as usual, some of them such as the Icelandic thorn character, `&thorn;`, `\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 LaTeX 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` ($\alpha$) `beta` ($\beta$) `gamma` ($\gamma$) `Gamma` ($\Gamma$)
- Integral and Sum. $\int$ grows large if the `bigint` package option is given.

  `int` ($\int$) `sum` ($\sum$)
- 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` ($\infty$) `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` ($\overbrace{\qquad}$) `underbrace` ($\underbrace{\qquad}$) and any (TeX) math accent name.

# 5 SGML Minimisation features

SGML (and hence HTML) support various minimisation features that aim to make it easier to enter the markp '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 `<li>` but the closing `</li>` at the end of the item may be omitted and inferred by the following `<li>` or `</ol>` 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 <img alt="DPC" src="dpc.gif">
```

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_O + CO_2_
</math>
<math class=chem>
H_2_O + CO_2_
</math>
<math box>
H_2_O + CO_2_
</math>
<math class=chem box>
H_2_O + CO_2_
</math>
```

$$H_2O + CO_2$$

$$\mathrm{H_2O} + \mathrm{CO_2}$$

$$\boxed{H_2O + CO_2}$$

$$\boxed{\mathrm{H_2O} + \mathrm{CO_2}}$$

## 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>
```

$$\displaystyle\int_3^1 \frac{1}{x+\frac{1}{x+\frac{2}{x+\frac{3}{x+\frac{4}{x}}}}}\,\mathrm{d}x$$

And the same integral with the standard integral sign.

$$\textstyle\int_3^1 \displaystyle\frac{1}{x+\frac{1}{x+\frac{2}{x+\frac{3}{x+\frac{4}{x}}}}}\,\mathrm{d}x$$

## 6.8 Oversized delimiters

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

$$\binom{1}{2}\Biggl(\begin{matrix}1\\2\end{matrix}\Biggr)$$

## 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>
```

$$\overbrace{abc}^{k} \quad \sqrt[3]{x}\sqrt{5} \quad \underline{abc}\widehat{abc}$$

## 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>
```

$$\begin{array}{cccc}
aa\ \text{first col} & \text{second col} & \text{third col} & \text{fourth col}bb \\
\text{row 2} & a_{22} & a_{23} & a_{24} \\
\text{row 3} & \multicolumn{2}{c}{a_{32}-a_{53}} & a_{34} \\
\text{row 4} & & & a_{44} \\
\text{row 5} & & & a_{54} \\
\text{row 6} & \multicolumn{1}{l}{al_{62}} & a_{63} & a_{64} \\
\text{row 7} & \multicolumn{1}{r}{ar_{72}} & a_{73} & a_{74}
\end{array}$$

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

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

$$aa \quad \begin{array}{cc}
\text{first col} & \begin{array}{ccc}\text{second col} & \text{third col} & \text{fourth col}\end{array} \\
\begin{array}{c}\text{row 2}\\ \text{row 3}\\ \text{row 4}\\ \text{row 5}\\ \text{row 6}\\ \text{row 7}\end{array} & \left(\begin{array}{ccc}
a_{22} & a_{23} & a_{24} \\
\multicolumn{2}{c}{a_{32}-a_{53}} & a_{34} \\
 & & a_{44} \\
 & & a_{54} \\
\multicolumn{1}{l}{al_{62}} & a_{63} & a_{64} \\
\multicolumn{1}{r}{ar_{72}} & a_{73} & a_{74}
\end{array}\right)
\end{array} bb$$

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>
```

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

## 6.11 Tables

HTML3 tables are not yet supported, but there is a minimal ammount 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

```
⟨*package⟩
```

```
\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.

```
\DeclareOption{netscape}
      {\def\HTML@not#1{\SGML@w{<#1> is not valid HTML}}}
```

```
\DeclareOption{html3}{\let\HTML@two@stop\relax}
```

```
\DeclareOption{nohyperref}{%
  \let\HTML@doname\@secondoftwo
  \def\HTML@dosrc#1#2{\emph{#2}}}
```

```
\DeclareOption{ftnhyperref}{%
  \let\HTML@doname\@secondoftwo
  \def\HTML@dosrc#1#2{\emph{#2}\footnote{HREF: \texttt{#1}}}}
```

```
\DeclareOption{hyperref}{%
  \AtBeginDocument{%
    \providecommand\href[2]{\special{html:<A href="#1">}%
                           #2\special{html:</A>}}%
    \providecommand\hypertarget[2]{\special{html:<A name="#1">}%
                           #2\special{html:</A>}}%
    \let\HTML@doname\hypertarget
    \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-existant command `typehtml`. Presumably this could be a batch file that calls netscape or some other WWW engine to process the URL.

```
\DeclareOption{dviwindo}{%
   \def\HTML@dosrc#1#2{{%
      \leavevmode\sbox\z@{#2}\count@\ht\z@\@tempcnta\wd\z@
      \if\string##\@car#1\@nil
        \special{button: \the\@tempcnta\space\the\count@\space
                 "\@gobble#1"}%
      \else
         \special{button: \the\@tempcnta\space\the\count@\space
              launch: typehtml "#1"}
      \fi
```

```
     \special{color push}\special{color rgb 0 1 0}%
     \unhbox\z@
     \special{color pop}}}%
   \def\HTML@doname#1#2{\leavevmode\special{mark: "#1"}#2}}%
\DeclareOption{imgalt}{}
\DeclareOption{imggif}{\SGML@w{img gif support not done yet}}
\DeclareOption{imgps}{\SGML@w{img ps support not done yet}}
\DeclareOption{smartquotedbl}{%
  \def\SGMLquotedbla{%
    \textquotedblleft\global\let\SGMLquotedbl\SGMLquotedblb}
  \def\SGMLquotedblb{%
    \textquotedblright\global\let\SGMLquotedbl\SGMLquotedbla}
  \let\SGMLquotedbl\SGMLquotedbla
  \let\SGML@savedeverypar\everypar
  \newtoks\everypar
  \SGML@savedeverypar{%
    \global\let\SGMLquotedbl\SGMLquotedbla\the\everypar}}
\DeclareOption{straightquotedbl}{%
  \DeclareTextCommandDefault{\textquotedbl}{{\ttfamily\char`\"}}%
  \let\SGMLquotedbl\textquotedbl}
\DeclareOption{chapter}{%
  \def\HTML@headings{%
    \chapter\section\subsection%
    \subsubsection\paragraph\subparagraph}}
\DeclareOption{chapter*}{%
  \def\HTML@headings{%
    {\chapter*}{\section*}{\subsection*}%
    {\subsubsection*}{\paragraph*}{\subparagraph*}}}
\DeclareOption{section}{%
  \def\HTML@headings{%
    \section\subsection%
    \subsubsection\paragraph\subparagraph\endgraf}}
\DeclareOption{section*}{%
  \def\HTML@headings{%
    {\section*}{\subsection*}%
    {\subsubsection*}{\paragraph*}{\subparagraph*}\endgraf}}
\DeclareOption{subsection}{%
  \def\HTML@headings{%
    \subsection%
    \subsubsection\paragraph\subparagraph\endgraf\endgraf}}
\DeclareOption{subsection*}{%
  \def\HTML@headings{%
    {\subsection*}%
    {\subsubsection*}{\paragraph*}{\subparagraph*}\endgraf\endgraf}}
\DeclareOption{bigint}{%
```

```
 \let\HTML@int\int
 \AtEndOfPackage{\RequirePackage{exscale}}}
```

```
\ExecuteOptions{section*,imgalt,html2,nohyperref,straightquotedbl}
```

```
\ProcessOptions
```

## 7.2 Fake SGML parser

```
\begingroup
```

```
\catcode‘\<=\active
\catcode‘\>=\active
\catcode‘\&=\active
\catcode‘\$=\active
\catcode‘\"=\active
\catcode‘\^=\active
\catcode‘\_=\active
\catcode‘\;=\active
\catcode‘\A=\active
\catcode‘\B=\active
\catcode‘\C=\active
\catcode‘\D=\active
```

```
\uccode‘\A=‘\{%
\uccode‘\B=‘\}%
```

```
\uccode‘\C=‘\|%
\uccode‘\D=‘\\%
```

```
\uppercase{\endgroup
```

\SGMLent@@

```
\def\SGMLent@@#1;{\csname SGML@E@#1\endcsname}
```

\SGMLent@@

```
\def\SGML@def@active#1>{%
 \expandafter\def\csname SGML@#1\endcsname}
```

```
\def\dohtml{%
 \begingroup
 \ifx;\@undefined\expandafter\let\expandafter;\string;\fi
 \ifx>\@undefined\expandafter\let\expandafter>\string>\fi
 \catcode‘\<=\active
 \catcode‘\>=\active
 \catcode‘\&=\active
 \catcode‘\{=\active
 \catcode‘\}=\active
 \catcode‘\$=\active
 \catcode‘\"=\active
 \catcode‘\^=\active
 \catcode‘\_=\active
 \catcode‘\\=\active
 \catcode‘\|=\active
 \catcode\endlinechar=10
 \catcode‘\%=12
 \catcode‘\#=12
 \catcode‘\;=\active
 \def\verbatim@nolig@list{\do\‘\do\,\do\’\do\-}
 \def<{\SGMLopen}%
 \def&{\SGMLent}%
 \let^\textasciicircum
 \let~\textasciitilde
 \def_{\_}%
 \let$\$%
 \def"{\SGMLquotedbl}%
 \def A{\{}%
 \def B{\}}%
 \def C{\texttt{|}}%
 \def D{\texttt{\char‘\\}}%
```

Need to be careful about writing to table of contents.

```
  \def\addcontentsline##1##2##3{%
    {\def<{\string<}\def&{\string&}%
      \addtocontents{##1}{\protect\dotochtml<html>}%
      \addtocontents{##1}{\protect\contentsline{##2}{##3}{\thepage}}%
      \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.

```
\def\dotochtml{%
 \dohtml
 \catcode‘\\\z@
 \catcode‘\{\@ne
  \catcode‘\}\tw@}
```

\SGMLshortend

```
\def\SGMLshortend{/}
```

\SGMLgrab@

```
\def\SGMLgrab@#1<#2>{%
  \edef\@tempd{\lowercase{\def\noexpand\@tempd{\gobblespc#2 \relax}}}%
  \@tempd
  \ifx\@tempd\SGMLshortend\let\@tempd\@tempc\fi
  \ifx\@tempb\@tempd
   \advance\@tempcnta\@ne
  \else
    \ifx\@tempc\@tempd
      \advance\@tempcnta\m@ne
    \fi
  \fi
  \ifnum\@tempcnta=\z@
```

```
    \expandafter\@tempa\expandafter{\the\@temptokena#1}%
  \else
    \addto@hook\@temptokena{#1<#2>}%
    \expandafter\SGMLgrab@
  \fi}
```

\SGMLopen

```
\def\SGMLopen#1>{%
  \SGMLopen@#1 \@nil}
```

```
}
```

\htmlinput

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

\gobblespc

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

\SGMLgrabber

```
\def\SGMLgrabber#1#2{%
  \def\@tempa{#2}%
  \@tempcnta\@ne
  \@temptokena{}%
  \lowercase{\def\@tempb{#1}\def\@tempc{/#1}}%
  \SGMLgrab@}
```

\SGMLopen@

```
\begingroup
\catcode`\"=\active
\uppercase{\endgroup
\def\SGMLopen@#1 #2\@nil{%
  \toks@{}%
  \edef\@tempa{\lowercase{\def\noexpand\SGMLelement{#1}}}\@tempa
  \if!\@car#1\relax\@nil
    \toks@{#1 #2}%
    \SGML@w{Declaration ignored\MessageBreak<\the\toks@>\MessageBreak}%
  \else
    \if$#2$\else
      \replacequotes#2"\@nil"%
      \SGMLafterfi
      \expandafter\toks@\expandafter{\expandafter}%
      \expandafter\SGMLgetattrib\the\toks@ \@nil
    \fi
    \expandafter\ifx\csname SGML@\SGMLelement
                          \expandafter\endcsname\relax
      \SGML@w{<\SGMLelement> undefined}%
    \else
      \csname SGML@\SGMLelement
        \expandafter\expandafter\expandafter\endcsname
    \fi
  \fi}
```

\replacequotes

```
\def\replacequotes#1"#2"{%
  \def\@tempb{#2}%
  \ifx\@tempb\@nnil
    \addto@hook\toks@{#1}%
  \else
    \addto@hook\toks@{#1{#2}}%
  \expandafter\replacequotes
  \fi}}
```

\SGMLafterfi

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

\SGMLgobbletofi

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

\SGMLgetattrib

```
\def\SGMLgetattrib#1 #2{%
  \ifx\box#1\box\else
    \SGMLgetval#1=$=\@nil
      \def\@tempa{#2}%
    \ifx\@tempa\@nnil
      \expandafter\SGMLgobbletofi
    \else
      \expandafter\SGMLafterfi
    \fi
    \SGMLgetattrib#2%
  \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.

```
\def\SGMLgetval#1=#2=#3\@nil{%
  \ifcat$#2%
    \lowercase{\SGML@addattrib\doimplied{#1}}%
  \else
    \lowercase{\SGML@addattrib{\do{#1}}}{#2}%
  \fi}
```

\SGML@addattrib

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

\SGML@w

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

\SGMLdef

```
\def\SGMLdef#1{%
  \ifcat\noexpand#1\noexpand~%
```

```
     \expandafter\SGML@def@active
  \else
     \expandafter\SGML@def
  \fi}
```

\SGMLdef make sure this is a catcode 12 >.

```
\edef\@tempa{\def\noexpand\SGML@def##1\string>}\@tempa{%
  \expandafter\def\csname SGML@#1\endcsname}
```

\SGMLent

```
\expandafter\def\expandafter\SGMLent\expandafter{%
  \expandafter\protect\csname& \endcsname}
```

&

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

\SGMLent@

```
\def\SGMLent@{%
 \ifx\@let@token\@sptoken
    \&%
 \else
    \expandafter\SGMLent@@
 \fi}
```

\SGMLentity

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

### 7.3 The HTML2 DTD

```
\SGMLdef<html>{}
\SGMLdef</html>{\@endhtml}
\let\@endhtml\endgroup
```

```
\SGMLdef<title>{\typeout{***TITLE***}\SGMLgrabber{title}\typeout}
```

```
\long\def\@tempa#1#2#3#4#5#6{%
  \SGMLdef<h1>{\SGMLgrabber{h1}{\HTMLsection{#1}}}%
  \SGMLdef<h2>{\SGMLgrabber{h2}{\HTMLsection{#2}}}%
  \SGMLdef<h3>{\SGMLgrabber{h3}{\HTMLsection{#3}}}%
  \SGMLdef<h4>{\SGMLgrabber{h4}{\HTMLsection{#4}}}%
  \SGMLdef<h5>{\SGMLgrabber{h5}{\HTMLsection{#5}}}%
  \SGMLdef<h6>{\SGMLgrabber{h6}{\HTMLsection{#6}}}}
```

```
\expandafter\@tempa\HTML@headings
```

```
\def\HTMLsection#1#2{#1{\ignorespaces#2\unskip}}
```

```
\SGMLdef<head>{}
\SGMLdef</head>{}
```

```
\SGMLdef<body>{}
\SGMLdef</body>{}

\SGMLdef<bodytext>{}
\SGMLdef</bodytext>{}

\SGMLdef<p>{\par}
\SGMLdef</p>{\par}

\SGMLdef<blockquote>{\begin{quote}}
\SGMLdef</blockquote>{\end{quote}}

\SGMLdef<address>{\begin{quote}}
\SGMLdef</address>{\end{quote}}

\SGMLdef<ul>{\begin{itemize}}
\SGMLdef</ul>{\end{itemize}}

\SGMLdef<ol>{\begin{enumerate}}
\SGMLdef</ol>{\end{enumerate}}

\SGMLdef<li>{\item}
\SGMLdef</li>{}

\SGMLdef<dl>{%
  \let\do\dldo
  \let\doimplied\dlimplied
  \begin{description}\the\toks@}
\SGMLdef</dl>{\end{description}}

\def\dldo#1#2{%
  \def\@tempa{compact}\def\@tempb{#1}%
  \ifx\@tempa\@tempb
    \itemsep\z@
    \advance\@totalleftmargin-\leftmargin
    \advance\linewidth\leftmargin
    \itemindent-\labelsep
    \leftmargin\z@
    \parshape \@ne \@totalleftmargin \linewidth
  \fi}
\def\dlimplied#1{\dldo{#1}\relax}

\def\itx#1{\item[#1]}
\SGMLdef<dt>{\begin{lrbox}\z@\bfseries\let\maybeenddt\enddt}
\SGMLdef</dt>{\maybeenddt}
\SGMLdef<dd>{\maybeenddt}
\SGMLdef</dd>{}
\def\enddt{\end{lrbox}\item[\unhbox\z@]}
\let\maybeenddt\relax

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

`\HTML@anchor` This handles the A tag.

```
\def\HTML@anchor#1{{%
  \let\@tempa\@gobble
  \def\_{\string_}%
  \let\do\ado
  \the\toks@
  \@tempa{#1}}}
```

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

```
\def\ado#1#2{%
  \def\@tempb{name}\def\@tempc{#1}%
  \ifx\@tempb\@tempc
    \let\@tempa\@firstofone
    \def\@tempa{\HTML@doname{#2}}%
  \else
    \def\@tempa{\HTML@dosrc{#2}}%
  \fi}
```

```
\SGMLdef<pre>{%
  \par
  \begingroup
  \parindent\z@
  \obeylines\verbatim@font\@noligs
  \frenchspacing\@vobeyspaces}
```

```
\SGMLdef</pre>{\endgroup}
```

```
\SGMLdef<tt>{\SGMLgrabber{tt}\texttt}
\SGMLdef<b>{\SGMLgrabber{b}\textbf}
\SGMLdef<i>{\SGMLgrabber{i}\textit}
\SGMLdef<em>{\SGMLgrabber{em}\emph}
\SGMLdef<strong>{\SGMLgrabber{strong}\textbf}
\SGMLdef<code>{\SGMLgrabber{code}\texttt}
\SGMLdef<samp>{\SGMLgrabber{samp}\textsf}
\SGMLdef<kbd>{\SGMLgrabber{kbd}\texttt}
\SGMLdef<var>{\SGMLgrabber{var}\textit}
\SGMLdef<cite>{\SGMLgrabber{cite}\textit}
```

```
\SGMLdef<form>{\par\medskip}
\SGMLdef</form>{\par\medskip}
```

```
\SGMLdef<select>{%
  \let\do\selectdo
  \the\toks@\par
  \begin{tabular}{|l|}%
  \hline\@tempc\\\hline
  \let\tabularnewline\relax
  \ignorespaces}
```

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

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

```
\SGMLdef<option>{%
  \gdef\optionbul{\phantom{$\bullet$}}%
  \let\do\optiondo
  \let\doimplied\optionimplied
  \the\toks@
  \tabularnewline
  \let\tabularnewline\\%
  \optionbul\space\ignorespaces}
\SGMLdef</option>{}
```

\optiondo Handle attributes to the OPTION element.

```
\def\optiondo#1#2{%
  \def\@tempa{selected}\def\@tempb{#1}%
  \ifx\@tempa\@tempb\gdef\optionbul{$\bullet$}\fi}
```

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

```
\def\optionimplied#1{%
  \def\@tempa{selected}\def\@tempb{#1}%
  \ifx\@tempa\@tempb\gdef\optionbul{$\bullet$}\fi}
```

```
\SGMLdef<input>{}
```

```
\SGMLdef<img>{{%
  \let\do\imgdo
  \def\@tempa{\doimage}%
  \the\toks@
  \@tempa}}
```

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

\imgdo Handle IMG attributes (not very usefully)

```
\def\imgdo#1{\csname img=#1\endcsname}
\expandafter\def\csname img=align\endcsname#1{%
  \SGML@w{align=#1 ignored}}
\expandafter\def\csname img=src\endcsname#1{%
  \SGML@w{src=#1 ignored}}
\expandafter\def\csname img=height\endcsname#1{%
  \SGML@w{height=#1 ignored}}
\expandafter\def\csname img=alt\endcsname#1{%
  \def\doimage{#1}}
```

Horizontal rules and line breaks.

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

These are obsolete in HTML3 but do them anyway.

```
\SGMLdef<xmp>{%
  \SGML@pre
  \def\@tempb{/xmp}%
  \let\SGMLopen\HTML@xmptest}
```

```
\SGMLdef<listing>{%
  \SGML@xmp
  \def\@tempb{/listing}}
```

```
\SGMLdef<plaintext>{%
  \SGML@xmp
  \def\@tempb{/plaintext}}%
```

\HTML@xmptest

```
\def\HTML@xmptest#1>{%
  \lowercase{\def\@tempa{#1}}%
  \ifx\@tempa\@tempb
    \endgroup
  \else
    \SGMLafterfi
    <#1>%
  \fi}
```

SGML syntax Character entities.

```
\SGMLentity{amp}{\&}
\SGMLentity{lt}{\ensuremath{<}}
\SGMLentity{gt}{\ensuremath{>}}
```

ISO Latin-1 Character entities.

```
\SGMLentity{aacute}{\'a}
\SGMLentity{Aacute}\'A{}
\SGMLentity{acirc}{\^a}
\SGMLentity{Acirc}{\^A}
\SGMLentity{agrave}{\`a}
\SGMLentity{Agrave}{\`A}
\SGMLentity{aring}{\r a}
\SGMLentity{Aring}{\r A}
\SGMLentity{atilde}{\~a}
\SGMLentity{Atilde}{\~A}
\SGMLentity{auml}{\"a}
\SGMLentity{Auml}{\"A}
\SGMLentity{aelig}{\ae}
\SGMLentity{AElig}{\AE}
\SGMLentity{ccedil}{\c c}
\SGMLentity{Ccedil}{\c C}
\SGMLentity{eth}{\dh}
\SGMLentity{ETH}{\DH}
\SGMLentity{eacute}{\'e}
\SGMLentity{Eacute}{\`E}
\SGMLentity{ecirc}{\^e}
\SGMLentity{Ecirc}{\^E}
\SGMLentity{egrave}{\`e}
\SGMLentity{Egrave}{\`E}
\SGMLentity{euml}{\"e}
\SGMLentity{Euml}{\"E}
\SGMLentity{iacute}{\'\i}
\SGMLentity{Iacute}{\'I}
\SGMLentity{icirc}{\^\i}
\SGMLentity{Icirc}{\^I}
```

```
\SGMLentity{igrave}{\`\i}
\SGMLentity{Igrave}{\`I}
\SGMLentity{iuml}{\"\i}
\SGMLentity{Iuml}{\"I}
\SGMLentity{ntilde}{\~n}
\SGMLentity{Ntilde}{\~N}
\SGMLentity{oacute}{\'o}
\SGMLentity{Oacute}{\'O}
\SGMLentity{ocirc}{\^o}
\SGMLentity{Ocirc}{\^O}
\SGMLentity{ograve}{\`o}
\SGMLentity{Ograve}{\`O}
\SGMLentity{oslash}{\oe}
\SGMLentity{Oslash}{\OE}
\SGMLentity{otilde}{\~o}
\SGMLentity{Otilde}{\~O}
\SGMLentity{ouml}{\"o}
\SGMLentity{Ouml}{\"O}
\SGMLentity{szlig}{\ss}
\SGMLentity{thorn}{\th}
\SGMLentity{THORN}{\TH}
\SGMLentity{uacute}{\'u}
\SGMLentity{Uacute}{\'U}
\SGMLentity{ucirc}{\^u}
\SGMLentity{Ucirc}{\^U}
\SGMLentity{ugrave}{\`u}
\SGMLentity{Ugrave}{\`U}
\SGMLentity{uuml}{\"u}
\SGMLentity{Uuml}{\"U}
\SGMLentity{yacute}{\'y}
\SGMLentity{Yacute}{\'Y}
\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 from . . . . . .[3] who 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.

```
\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 `<font>`. (I thought these were Netscape-specific but the technical notes at Spyglass's web site showed that I was

[3] Name withheld to protect the guilty

wrong. [mjd,1996/03/20])

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

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

```
\SGMLdef<font>{\HTML@not{font}\begingroup
  \let\do\fontdo\the\toks@}
\SGMLdef</font>{\endgroup}
```

`\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.

```
\def\fontdo#1#2{%
  \def\@tempa{size}\def\@tempb{#1}%
  \ifx\@tempa\@tempb
    \font@switch#2\relax\@nil
  \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.

```
\toks@\expandafter{\set@fontsize{#1}{#2}{#3}}
\edef\@tempa{%
  \def\noexpand\set@fontsize##1##2##3{\the\toks@\noexpand\set@fontnum}}
\@tempa
```

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

```
\def\set@fontnum{\dimen@\f@size\p@
  \dimen@\mul@ptsize\dimen@
  \count@\dimen@ \divide\count@\p@
  \advance\count@ -5\relax
  \edef\@fontnum{\number\count@}}
```

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

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

5 = `\normalsize` I think

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

Initialize `\mul@ptsize`

```
\ifcase 0\@ptsize\relax
  \global\let\mul@ptsize\@empty% case 0, ptsize = 10
  \or\gdef\mul@ptsize{.9091}%    case 1, ptsize = 11
  \else\gdef\mul@ptsize{.8333}%  case 2, ptsize = 12
\fi
```

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

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

```
\fi
```

## 7.5 The HTML3 DTD

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

```
\HTML@two@stop
\SGML@w{HTML3 support not finished yet}
```

```
\SGMLdef<math>{\SGMLgrabber{math}\domath}
```

```
\SGMLdef<sup>{^\bgroup\HTMLscriptmap}
\SGMLdef</sup>{\egroup}
```

```
\SGMLdef<sub>{_\bgroup\HTMLscriptmap}
\SGMLdef</sub>{\egroup}
```

GRUMBLE! GRUMBLE! GRUMBLE! Possibly the worst feature of TEX'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...

```
\SGMLdef<box>{\SGMLgrabber{box}\dobox}
```

```
\begingroup
\catcode‘\<=\active
\catcode‘\>=\active
\catcode‘\&=\active
\catcode‘\_=\active
\catcode‘\^=\active
\catcode‘\"=\active
```

`\domath` Handle the MATH element. The body is pre-expanded one level to replace `{ }` by BOX elements, and to replace any SGML entitity references by single TEX 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.

```
\gdef\domath#1{%
 {{\def&{\expandafter\expandafter\expandafter\noexpand\SGMLent@@}%
  \let<\relax\let>\relax\let_\relax\let^\relax\let"\relax
  \def\{{<box>}\def\}{</box>}%
  \xdef\@gtempa{#1}}%
 \let\do\mathdo
 \let\doimplied\mathimplied
 \the\toks@
 \[%
 \m@th\nulldelimiterspace\z@
 \def^{<sup>}%
 \def_{<sub>}%
 \@gtempa\]}}
```

`\HTMLscriptmap` Set up the shortref map used in super and subscripts.

```
\gdef\HTMLscriptmap{%
```

```
  \def^{</sup>}%
  \def_{</sub>}}
```

\dobox Handle the BOX element. First deal with the attributes, then set up the shortref map. Then start looking for a LEFT tag.

```
\gdef\dobox#1{%
 {\let\do\boxdo
   \let\bigstrut\relax
   \the\toks@
   \def^{<sup>}%
   \def_{<sub>}%
   \lookleft@#1<left>\@nil}}
```

\lookleft@ See whether this BOX element contains a LEFT tag. Supply a 'null delimiter' if not one supplied.

```
\gdef\lookleft@#1<left>#2\@nil{%
  \if$#2$%
    {\left.\bgroup#1\mayberight}%
  \else
     \lookbox@#1<box>\@nil#2\@nil
  \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.

```
\gdef\lookbox@#1<box>#2\@nil#3<left>\@nil{%
  \if$#2$%
     {\maybeleft#1\@nil#3\mayberight}
  \else
     {#1 \boxtofront#2 <left> #3}%
  \fi}
```

\boxtofront After all that messing around need to put the BOX tag back where we found it.

```
\gdef\boxtofront#1<box>{<box>#1}
```

```
\endgroup
```

\mathdo

```
\def\mathdo#1#2{%
  \def\@tempa{class-chem}\def\@tempb{#1-#2}%
  \ifx\@tempa\@tempb
     \everymath{\fam\z@}\everydisplay{\fam\z@}%
  \fi}
\def\mathimplied#1{%
  \def\@tempa{box}\def\@tempb{#1}%
  \ifx\@tempa\@tempb
     \def\[{\center\setbox\z@\hbox\bgroup$\displaystyle}%
     \def\]{$\egroup\fbox{\box\z@}\endcenter}%
  \fi}
```

\boxdo

```
\def\boxdo#1#2{%
  \def\@tempa{size}\def\@tempb{#1}%
  \ifx\@tempa\@tempb
     \def\@tempb{#2}
     \def\@tempa{normal}\ifx\@tempa\@tempb\def\@tempc{1}\fi
     \def\@tempa{medium}\ifx\@tempa\@tempb\def\@tempc{2}\fi
     \def\@tempa{large}\ifx\@tempa\@tempb\def\@tempc{3}\fi
     \def\@tempa{huge}\ifx\@tempa\@tempb\def\@tempc{4}\fi
     \edef\bigstrut{\vrule\@height\@tempc\ht\strutbox\@width\z@}
  \fi}
```

\SGML@left

```
\SGMLdef<left>{\left.\bgroup}
```

\mayberight

```
\def\mayberight{\egroup\bigstrut\right.}
```

\maybeleft

```
\def\maybeleft#1#2\@nil{%
   \in@{#1}{()[]\SGML@E@rbrace\SGML@E@lbrace}%
   \ifin@
     \left#1\bgroup#2%
   \else
      \let\SGML@E@int\HTML@bigint
      #1#2\left.\bgroup\let\SGML@E@int\int
   \fi}
```

\righttest

```
\def\righttest#1{%
   \in@{#1}{()[]\SGML@E@rbrace\SGML@E@lbrace}%
   \ifin@
     \right#1\let\mayberight\relax
   \else
      \right.\let\mayberight\relax\expandafter#1%
   \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

```
\ifx\HTML@int\@undefined
```

\HTML@bigint Normally just use the standard `\int`.

```
\let\HTML@bigint\int
```

```
\else
```

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

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

\HTML@bigint

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

\HTMLafterbigint

```
\def\HTMLafterbigint{%
   \dimen@.5\ht\z@
   \advance\dimen@.5\dp\z@
   {\SGMLdef<sup>{\setbox\tw@\hbox\bgroup\HTMLscriptmap$\scriptstyle}%
    \SGMLdef<sub>{\setbox4\hbox\bgroup\HTMLscriptmap$\scriptstyle}%
    \SGMLdef</sup>{$\egroup}%
    \SGMLdef</sub>{$\egroup}%
    \setbox\tw@\box\voidb@x
    \setbox4\box\voidb@x
    \@tempa
    \ifdim\dimen@>\f@size\p@
```

At this point, could do \fontsize\dimen@\z@\selectfont but that would load *all* the math fonnts 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).

```
       \mathop{\hbox{\DeclareFixedFont\@tempa{OMX}{cmex}{m}{n}\dimen@
           $\displaystyle\textfont\thr@@\@tempa\HTML@int$}}%
    \else
       \HTML@int
    \fi
    }\left.\box\z@}
```

```
\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.

```
\SGMLdef<over>{\over}
\SGMLdef<atop>{\atop}
\SGMLdef<choose>{\choose}
```

```
\SGMLdef<right>{\egroup\bigstrut\righttest}
```

```
\SGMLdef<above>{\SGMLgrabber{above}%
    {\let\@tempc\overlineop
    \let\do\abovedo
    \the\toks@
    \@tempc}}
```

```
\SGMLdef<below>{\SGMLgrabber{below}%
    {\let\@tempc\underlineop
    \let\do\abovedo
    \the\toks@
    \@tempc}}
```

```
\def\overlineop#1{\mathop{\overline{#1}}}
\def\underlineop#1{\mathop{\underline{#1}}}
```

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

```
\SGMLdef<vec>{\SGMLgrabber{vec}\vec}
\SGMLdef<bar>{\SGMLgrabber{bar}\bar}
\SGMLdef<dot>{\SGMLgrabber{dot}\dot}
\SGMLdef<ddot>{\SGMLgrabber{ddot}\ddot}
\SGMLdef<hat>{\SGMLgrabber{hat}\hat}
\SGMLdef<tilde>{\SGMLgrabber{tilde}\tilde}
```

```
\SGMLdef<t>{\SGMLgrabber{t}\mathrm}
\SGMLdef<bt>{\SGMLgrabber{bt}\mathbf}
```

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

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

I think the HTML3 DTD is wrong here[4], it allows the OF element to take content, which is at variance with the description in the text.

```
\SGMLdef<of>{\egroup$\egroup\SGMLgrabber{root}\offudge}
\SGMLdef</of>{}
\def\offudge#1{\mathpalette\r@@t{#1}}
```

```
\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.

```
\newcount\HTMLrow
\newcount\HTMLcol
\newcount\HTMLrowspan
\newcount\HTMLcolspan
```

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

[4] Since confirmed by Dave Raggett, the HTML3 author

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

```
\SGMLdef<array>{{\ifnum`}=0\fi
  \let\do\arraydo
  \let\doimplied\arrayimplied
  \let\HTMLal.%
  \let\HTMLar.%
  \global\HTMLrow\z@
  \let\HTMLabox\vcenter
  \the\toks@
  \setbox\z@\vbox\bgroup\halign\bgroup
    \strut\span\HTMLacolspec\cr\nocr}
```

`\HTMLacolspec`

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

`\HTMLamakepream`

```
\def\HTMLamakepream#1{%
  \let\HTMLacolspec\@empty
  \let\@sharp\relax
  \lowercase{\@tfor\@tempc:=#1}\do{%
    \if\@tempc l%
      \edef\HTMLacolspec{\HTMLacolspec\@sharp\hfill&}%
    \else
      \if\@tempc c%
        \edef\HTMLacolspec{\HTMLacolspec\hfill\@sharp\hfill&}%
      \else
        \if\@tempc r%
          \edef\HTMLacolspec{\HTMLacolspec\hfill\@sharp&}%
        \else
          \if\@tempc +%
            \edef\HTMLacolspec{\HTMLacolspec$+$}%
          \else
            \if\@tempc -%
              \edef\HTMLacolspec{\HTMLacolspec$-$}%
            \else
              \if\@tempc =%
                \edef\HTMLacolspec{\HTMLacolspec$=$}%
              \fi
            \fi
          \fi
        \fi
      \fi
    \fi}%
  \def\@sharp{########}%
  \edef\HTMLacolspec{\HTMLacolspec&\@sharp}}
```

```
\SGMLdef</array>{\HTMLendarray}
```

```
\let\HTMLcr\cr
```

\HTMLendarray Non LABELS ending

```
\def\HTMLendarray{%
  \endi\crcr\egroup\egroup
  \ifx\HTMLabox\vtop
    \setbox\z@\vtop{\unvbox\z@}%
  \else
     \ifx\HTMLabox\vcenter
        \dimen@\ht\z@
         \advance\dimen@\dp\z@
         \divide\dimen@\tw@
         \advance\dimen@-\ht\z@
        \setbox\z@\hbox{\raise\dimen@\box\z@}%
    \fi
  \fi
  \dimen@=\ht\z@
  \setbox\z@
\hbox{$\left\HTMLal\kern-1em\vcenter{\box\z@}\kern-1em\right\HTMLar$}%
  \advance\dimen@-\ht\z@
  \raise\dimen@\box\z@
  \ifnum`{=0\fi}}
```

\HTMLendarraylabels LABELS ending

```
\def\HTMLendarraylabels{%
  \endi\crcr\strut\cr\egroup\egroup
  \setbox2=\vsplit\z@ to \baselineskip
  \setbox\z@\vbox{\unvbox\z@\global\setbox\@ne\lastbox}%
  \setbox4\hbox{\unhbox\@ne\unskip\global\setbox\@ne\lastbox}%
  \vcenter{%
  \box2
  \hbox{$\kern\wd\@ne
     \left\HTMLal\kern-\wd\@ne
       \vcenter{\box\z@}%
     \right\HTMLar$}}%
  \ifnum`{=0\fi}}
```

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

```
\SGMLdef<row>{%
  \endi\HTMLcr
  \global\advance\HTMLrow\@ne
  \global\HTMLcol\z@}
```

```
\SGMLdef</row>{}
```

```
\SGMLdef<item>{%
  \let\do\itemdo
  \gdef\@gtempa{\global\advance\HTMLcol\@ne}%
  \gdef\@gtempb{}%
  \gdef\@gtempc{}%
  \global\HTMLcolspan\@ne
  \the\toks@
  \endi%
  \@gtempc
  \@gtempa
```

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

```
  \spanifneeded
```

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

```
  \@gtempb
```

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

```
  \setbox\z@\hbox\bgroup$%
  \def\endi{\unskip$\egroup%
     \quad\HTMLaleft\box\z@\HTMLaright\quad&}%
   \ignorespaces}
```

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

```
\def\spanifneeded{%
  \edef\@tempa{\noexpand\in@{,\the\HTMLrow/\the\HTMLcol,}{\spanitems}}%
  \@tempa
  \ifin@
     \@firstofone{&}\global\advance\HTMLcol\@ne
   \expandafter\spanifneeded
  \fi}
```

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

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

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

```
\let\HTMLaleft\hfil
\let\HTMLaright\hfil
```

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

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

\arraydo Handle ARRAY attributes.

```
\def\arraydo#1#2{%
  \def\@tempa{#1}\def\@tempb{#2}%
  \def\@tempc{align}%
  \ifx\@tempa\@tempc
    \def\@tempc{top}%
    \ifx\@tempb\@tempc
      \let\HTMLabox\vtop
     \else
      \def\@tempc{bottom}%
      \ifx\@tempb\@tempc
        \let\HTMLabox\vbox
      \fi
    \fi
  \else
    \def\@tempc{ldelim}%
    \ifx\@tempa\@tempc
      \let\HTMLal\@tempb
    \else
      \def\@tempc{rdelim}%
      \ifx\@tempa\@tempc
        \let\HTMLar\@tempb
      \else
        \def\@tempc{labels}%
        \ifx\@tempa\@tempc
          \let\HTMLendarray\HTMLendarraylabels
        \else
          \def\@tempc{colspec}%
          \ifx\@tempa\@tempc
            \HTMLamakepream{#2}%
          \fi
        \fi
      \fi
    \fi
  \fi}
```

`\arrayimplied`

`\itemdo` Handle ITEM attributes

```
\def\itemdo#1#2{%
  \def\@tempa{#1}\def\@tempb{#2}%
  \def\@tempc{colspan}%
  \ifx\@tempa\@tempc
    \global\HTMLcolspan#2\relax
    \gdef\@gtempa{\@multispan#2\relax\global\advance\HTMLcol#2\relax}%
  \else
    \def\@tempc{align}%
    \ifx\@tempa\@tempc
      \def\@tempc{left}%
      \ifx\@tempb\@tempc
        \gdef\@gtempb{\let\HTMLaleft\relax}%
      \else
        \def\@tempc{right}%
        \ifx\@tempb\@tempc
          \gdef\@gtempb{\let\HTMLaright\relax}%
        \fi
      \fi
    \else
      \def\@tempc{rowspan}%
      \ifx\@tempa\@tempc
```

```
        \global\HTMLrowspan#2\relax
        \gdef\@gtempc{%
          \@tempcnta=\HTMLrow
          \advance\@tempcnta\HTMLrowspan
```

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

```
          \loop
            \@tempcntb=\HTMLcol
            \advance\@tempcntb\HTMLcolspan
            \advance\@tempcnta\m@ne
            \ifnum\@tempcnta>\HTMLrow
            {\loop
               \xdef\spanitems{%
                  \spanitems\the\@tempcnta/\the\@tempcntb,}%
               \advance\@tempcntb\m@ne
             \ifnum\@tempcntb>\HTMLcol
             \repeat}%
          \repeat}%
      \fi
    \fi
  \fi}
```

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

```
\def\spanitems{,}
```

```
\SGMLentity{thinsp}{\,}
\SGMLentity{emsp}{\quad}
```

Far from final list of math symbol entity names…

```
\SGMLentity{alpha}{\alpha}
\SGMLentity{beta}{\beta}
\SGMLentity{gamma}{\gamma}
\SGMLentity{Gamma}{\Gamma}

\SGMLentity{int}{\int}
\SGMLentity{sum}{\sum}

\SGMLentity{lbrace}{\lbrace}
\SGMLentity{rbrace}{\rbrace}

\SGMLentity{times}{\times}
\SGMLentity{cup}{\cup}
\SGMLentity{cap}{\cap}
\SGMLentity{vee}{\vee}
\SGMLentity{wedge}{\wedge}
\SGMLentity{infty}{\infty}
\SGMLentity{oplus}{\oplus}
\SGMLentity{ominus}{\ominus}
\SGMLentity{otimes}{\otimes}

\SGMLentity{sin}{\sin}
\SGMLentity{cos}{\cos}
\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.

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

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

```
\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?

```
\SGMLdef<caption>{\end{tabular}\begingroup\bfseries}
\SGMLdef</caption>{\endgroup\par\smallskip\begin{tabular}{*{10}{c}}}
```

```
⟨/package⟩
```