Reinhold Kainhofer

The CSSSave Package for Mathematica
Extending the built-in HTMLSave function with (cascading) style sheets

Graz University of Technology
Deltasoft mathematics

Zagreb, September 26, 2003
Motivation

• Style sheets give Mathematica a stylish look

• But converted HTML pages look boring (black text on white bg., no colors, borders, spacing, layout)
  Only some (more or less advanced) HTML markup, hardcoded in Mathematica’s Style Sheet

• HTML supports style sheets as well: Cascading Style Sheets (CSS) as defined by the W3C

Problem:
Try to extend the existing HTMLSave function of Mathematica
No duplication, expansion of the function

Solution: CSSSave` Package (http://csssave.sf.net)
How to work with the package

- Loading: `<<CSSSave`
- Converting to html: HTMLSave will automatically use the package
- Generating a Cascading style sheet from a notebook: CSSSave function (similar to HTMLSave)
  `CSSSave[toFile_String, nb_Notebook, opts___?OptionQ]`
- In Mathematic 5.0 this functionality is already included. The extra package is not needed.
Style Sheets

- Mathematica uses its own style sheets in Mathematica syntax, no inheritance
- HTML uses Cascading Style Sheets, even inheritance is possible

<table>
<thead>
<tr>
<th>Mathematica</th>
<th>Cascading Style Sheet (CSS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background→RGBColor[0, 1, 0]</td>
<td>background-color : rgb(0%, 100%, 0%);</td>
</tr>
<tr>
<td>FontColor→GrayLevel[1]</td>
<td>color : rgb(100%, 100%, 100%);</td>
</tr>
<tr>
<td>FontFamily→”Lucida”</td>
<td>font-family : ”Lucida”;</td>
</tr>
<tr>
<td>CellFrame→{0, 0}, {0, 0.25}</td>
<td>border-style : solid none none none none;</td>
</tr>
<tr>
<td>CellMargins→{36, 20}, {10, 20}</td>
<td>border-width : 1px 0px 0px 0px;</td>
</tr>
<tr>
<td>CellFrameMargins→{10, 4}, {6, 2}</td>
<td>margin : 20px 20px 10px 36px;</td>
</tr>
<tr>
<td>TextAlignment→Left</td>
<td>padding : 2px 4px 6px 10px;</td>
</tr>
<tr>
<td>FontVariations→\”Underline”→True}</td>
<td>text-align : Left;</td>
</tr>
<tr>
<td>CellDingbat→”[FilledCircle]”</td>
<td>text-decoration : underline;</td>
</tr>
<tr>
<td></td>
<td>display: list-item; list-style-type: square;</td>
</tr>
</tbody>
</table>
How to intercept the HTML export (menu item)

• MenuItem calls \texttt{FrontEnd\textasciigrave{}DoHTMLSave[]}, which is a wrapper for \texttt{HTMLSave}

• \Rightarrow Write our own \texttt{HTMLSave} function, force Mathematica to use that one

• This can be done using a more specific pattern with a condition:

\begin{verbatim}
$UseCSSInternal = True;
HTMLSave[ destinPath:(_String|...), ..., 
    convOpts___?OptionQ] /; $UseCSSInternal :=
Block[{$UseCSSInternal=False},
    (* Here comes our own conversion code. We can even call HTMLSave here, and the internal one will be called *)
]
\end{verbatim}
General structure of our own HTMLSave function

```plaintext
HTMLSave[destinPath:(_String|FileName|FrontEnd`FileName), nb_Notebook,
         convOpts___?OptionQ]/; ($UseCSSInternal) /;
(Head[$FrontEnd] === FrontEndObject) :=
Module[{res, newConvOpts},
   (* Convert the styles to css *)
   newConvOpts = CSSSave[destinPath, nb, convOpts];
   (* Prepare the arguments for the internal HTMLSave function *)
   Block[{$UseCSSInternal = False},
      res = HTMLSave[ destinPath, nb, newConvOpts ]
   ];
   (* Clean up if necessary *)
   res
];
```
Converting the style sheets

• Extract all style cells by Cases, then generate a string from each.

• Convert each Cell[StyleData[style_String], opts___?OptionQ] to the form

```plaintext
   stylename, .style {
      propertyname: propertyvalue;
      ...
   }
```

Generate arguments for HTMLSave

• Use the MarkupRules option of HTMLSave. Gives conversion rules for each style.

• Need rules for inline text and block level elements, e.g.:

```
"Text" -> {{"<span class="Text"">", ", "</span>"},
{"<p class="Text"">", ", "</p>"}}
```

The rest is done by the internal HTMLSave
Results: Using the CSSSave package

The same notebook in Mathematica and converted HTML using the CSSSave package.
Results: The old HTMLSave results

Conversion result with the built-in HTMLSave.

Either only graphics are created (no copying), or all formatting is lost.