Use Cascading Stylesheets to display XML

Presented by developerWorks, your source for great tutorials

ibm.com/developerWorks

Table of contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

Section 1. Tutorial introduction

Who should take this tutorial?

Most uses of XML eventually result in some sort of Web browser output. Today the most common approach to preparing information in XML for Web publishing is to use XSLT or a lower-level programming language to convert the XML to HTML. However, in some cases you can achieve easier XML Web publishing by using Cascading Stylesheets (CSS) to instruct Web browsers on how to directly display XML. CSS is best known as the recommended method for specifying the presentation of HTML, but recent versions also support XML, and have been embraced by browser vendors as the best-supported means of displaying XML.

Anyone who works with XML should take this tutorial. Even if CSS doesn't cover your needs for production Web publishing, it is a great tool for debugging and experimentation.

Prerequisites

This tutorial assumes knowledge of XML, and some basic knowledge of CSS. If you aren't familiar with XML, I recommend you first take the "*Introduction to XML*" tutorial here on developerWorks.

I recommend, but do not require, familiarity with XML Namespaces and XSLT. If you aren't familiar with XSLT, you might want to take the tutorial "*Create multi-purpose Web content with XSLT*," also here on developerWorks.

I highly recommend that you try out the examples. To do so you will need a Web browser that supports XML and CSS Level 2 or better. I present the output of all examples using Firefox 1.0 Preview Release on Linux. *Firefox* (http://www.getfirefox.com/) is the popular Web browser available on Windows, Mac OS X, Linux, and other platforms. It is based on Mozilla's rendering engine, which has always been known as a very CSS-compliant browser.

About the XML and CSS examples in this tutorial

In this tutorial you will see many examples of CSS files. All the files used in this tutorial are in the zip file, x-xmlcss-tutorial-files.zip. In this package, all files start with a prefix indicating the section that covers them and the order the examples appear within the section. For example, files from the first example in the third section are named starting with "eg_3_1".

Files with the .css extension are Cascading Stylesheets, and files with the .xml extension are sample XML documents that you can display using the CSS of the same prefix. All but one of the XML documents are in substance the same as eg_3_1.xml, but with different instructions for loading each CSS file.

I do list the example files in each panel, so you can easily locate and experiment with the examples as you take this tutorial.

About the author

Uche Ogbuji is a consultant and co-founder of *Fourthought Inc.* (http://Fourthought.com) , a software vendor and consultancy specializing in XML solutions for enterprise knowledge management. Fourthought develops *4Suite*, an open source platform for XML, RDF, and knowledge-management applications. Mr. Ogbuji is also a lead developer of the *Versa* (http://uche.ogbuji.net/tech/rdf/versa/) RDF query language. He is a computer engineer and writer born in Nigeria, living and working in Boulder, Colorado, USA. You can contact Mr. Ogbuji at *uche.ogbuji@fourthought.com*.

Section 2. Understanding where CSS comes in

The problems to be solved

CSS can prove useful in any scenario where you need to display XML directly and with little fuss. It might be the primary means of such display, or just a quick way to stage, preview, debug, or analyze content that will eventually go through more sophisticated workflow.

In illustrating how easy it is to display XML using CSS, this tutorial works around a sample scenario. The Tech Society is a hypothetical organization that publishes technical papers and reports submitted by diverse authors. The articles are submitted in a simple XML format. Tech Society editors require a lot of flexibility when they display the articles for author preview and actual publication.

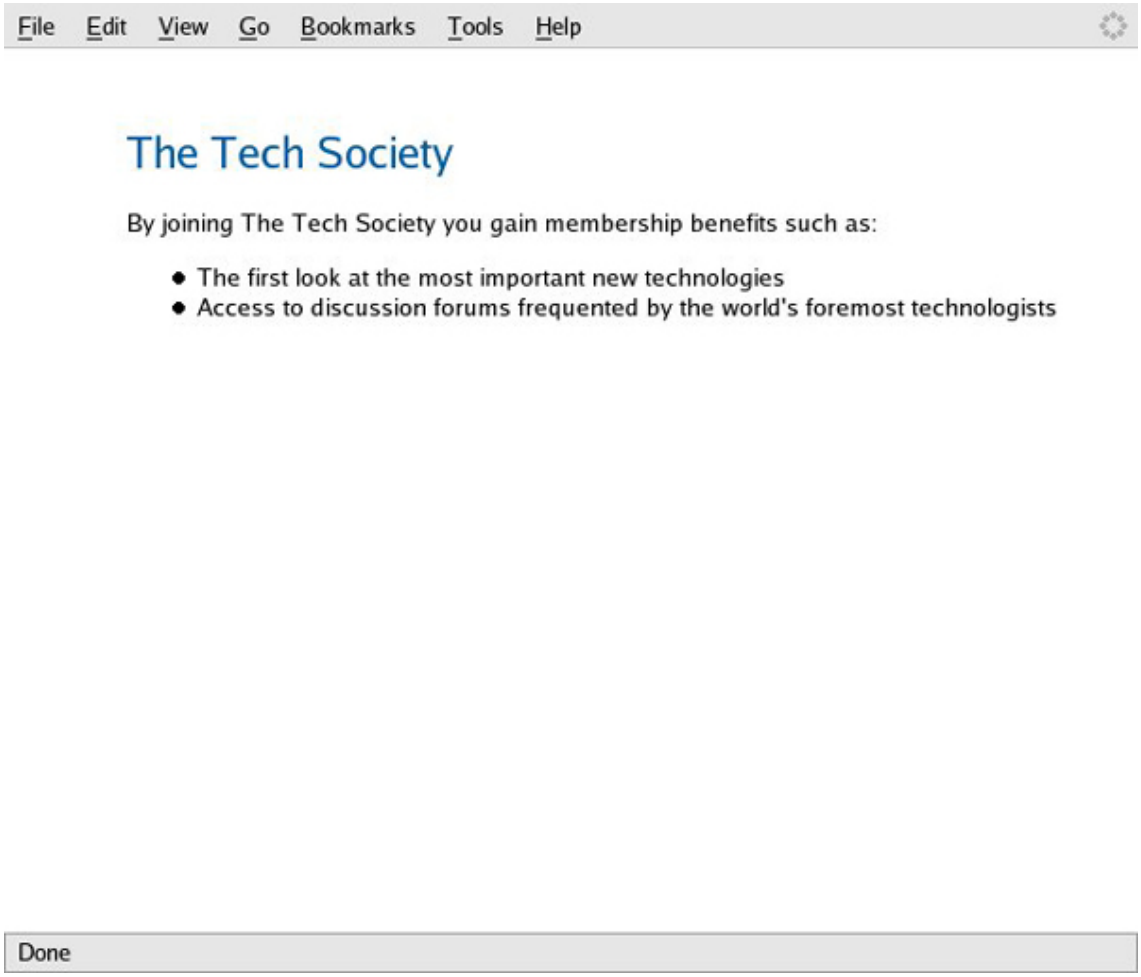
This tutorial uses a lot of stylesheet examples within this scenario to demonstrate the usefulness of CSS for XML display. To get the most out of it, experiment with the examples in your favorite XML-aware browser.

Using XHTML with CSS

The point of this tutorial is to teach you how to style any XML at all using CSS, but XHTML is worth a special note. Most browsers treat XHTML much like HTML and this means they apply their default CSS rules to XHTML. If you want to use your own stylesheet, do so using the usual HTML methods, either inserting the rules into `style` elements, or through a `link` with `rel="stylesheet"` (there are fancier ways, including JavaScript magic, but I won't go into those here). As you'll soon see, the W3C does recommend -- and all XML-aware browsers implement -- an `xml-stylesheet` processing instruction for linking stylesheets to general XML vocabularies, but XHTML is a special case for which I don't recommend `xml-stylesheet`. The following XHTML (eg_2_2.xhtml in the x-xmlcss-tutorial-files.zip), demonstrates HTML-like stylesheet linking in order to specialize the look and feel. The image below shows the visual result, which you may find somewhat familiar if you've ever browsed a W3C page.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>The Tech Society</title>
  <link rel="stylesheet" type="text/css"
        href="http://www.w3.org/StyleSheets/TR/base.css"/>
</head>
<body>
  <h1>The Tech Society</h1>
  <p>
```

```
    By joining The Tech Society you gain membership benefits such as:
    </p>
    <ul>
      <li>The first look at the most important new technologies</li>
      <li>Access to discussion forums frequented by the world's
          foremost technologists</li>
    </ul>
  </body>
  </html>
```



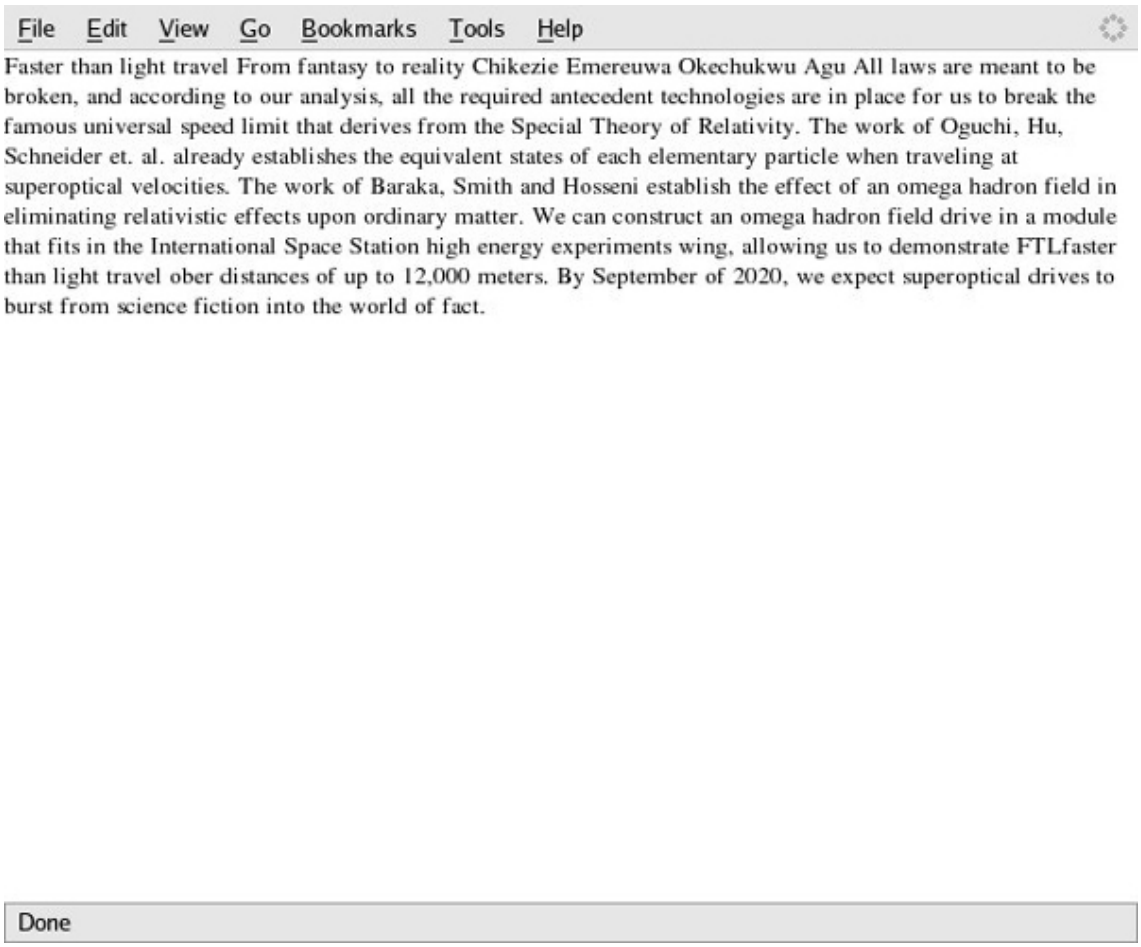
Default behavior displaying XML without any stylesheet

Any browser that can use stylesheets for XML can read XML perfectly well without any stylesheet at all. Generally, though, the resulting display is not very friendly. The following is a sample document from later on in this tutorial.

The image below shows the result of viewing this in Firefox. As you can see, all the content from all the elements is crammed into one big chunk and it's hard to

make much sense of it. Most browsers render XML by ignoring tags and attributes, and simply dumping content as plain text in document order. You will soon learn how to remedy this situation.

```
<paper>
  <prologue>
    <title>Faster than light travel</title>
    <subtitle>From fantasy to reality</subtitle>
    <author member='yes' e-mail="cemereuwa@nasa.gov">Chikezie Emereuwa</author>
    <author member='no' e-mail="okey.agu@navy.mil">Okechukwu Agu</author>
    <main-contact e-mail='cemereuwa@nasa.gov'/>
    <abstract>All laws are meant to be broken, and according
    to our analysis, all the required antecedent technologies are in place
    for us to break the famous universal speed limit that derives from
    the Special Theory of Relativity.</abstract>
  </prologue>
  <section>
    <para>The work of Oguchi, Hu, Schneider et. al. already establishes
    the equivalent states of each elementary particle when traveling at
    superoptical velocities.  The work of Baraka, Smith and Hosseni establish
    the  effect of an omega hadron field in eliminating relativistic effects
    upon ordinary matter.</para>
  </section>
  <section>
    <para>We can construct an omega hadron field drive in a module
    that fits in the International Space Station high energy experiments wing,
    allowing us to demonstrate lt;abbreviation><short>FTL</short><long>faster
    than light</long></abbreviation> travel over distances
    of up to 12,000 meters.</para>
    <para>By September of 2020, we expect superoptical drives to burst from
    science fiction into the world of fact.</para>
  </section>
</paper>
```


File Edit View Go Bookmarks Tools Help
Faster than light travel From fantasy to reality Chikezie Emereuwa Okechukwu Agu All laws are meant to be broken, and according to our analysis, all the required antecedent technologies are in place for us to break the famous universal speed limit that derives from the Special Theory of Relativity. The work of Oguchi, Hu, Schneider et. al. already establishes the equivalent states of each elementary particle when traveling at superoptical velocities. The work of Baraka, Smith and Hosseni establish the effect of an omega hadron field in eliminating relativistic effects upon ordinary matter. We can construct an omega hadron field drive in a module that fits in the International Space Station high energy experiments wing, allowing us to demonstrate FTLfaster than light travel ober distances of up to 12,000 meters. By September of 2020, we expect superoptical drives to burst from science fiction into the world of fact.
Done

Section 3. Select and style XML elements

The display property

Web browsers know the semantics of HTML elements, and in particular how to display each element based on the relevant specification. Users can override the default display using CSS, but that's unnecessary to get a useful display of any HTML element. In the case of XML, the browser generally doesn't know the semantics of elements (XHTML is one big exception), and the usual default, demonstrated in Default behavior displaying XML without any stylesheet on page 5, is rarely very useful. As a foundation, the browser has to be told the most basic information on how to display each element. You do this by applying a CSS `display` property to a selector that addresses the element of interest. You don't often use this property in CSS for HTML, so you may not be familiar with it, but the first value of `display` to learn is `block`, which instructs the browser to use white space to separate the element content from other material, just as it would for HTML `p` or `div` tags.

The following CSS (eg_3_1.css in the x-xmlcss-tutorial-files.zip) instead instructs the browser to display the content of each element in a distinct block.

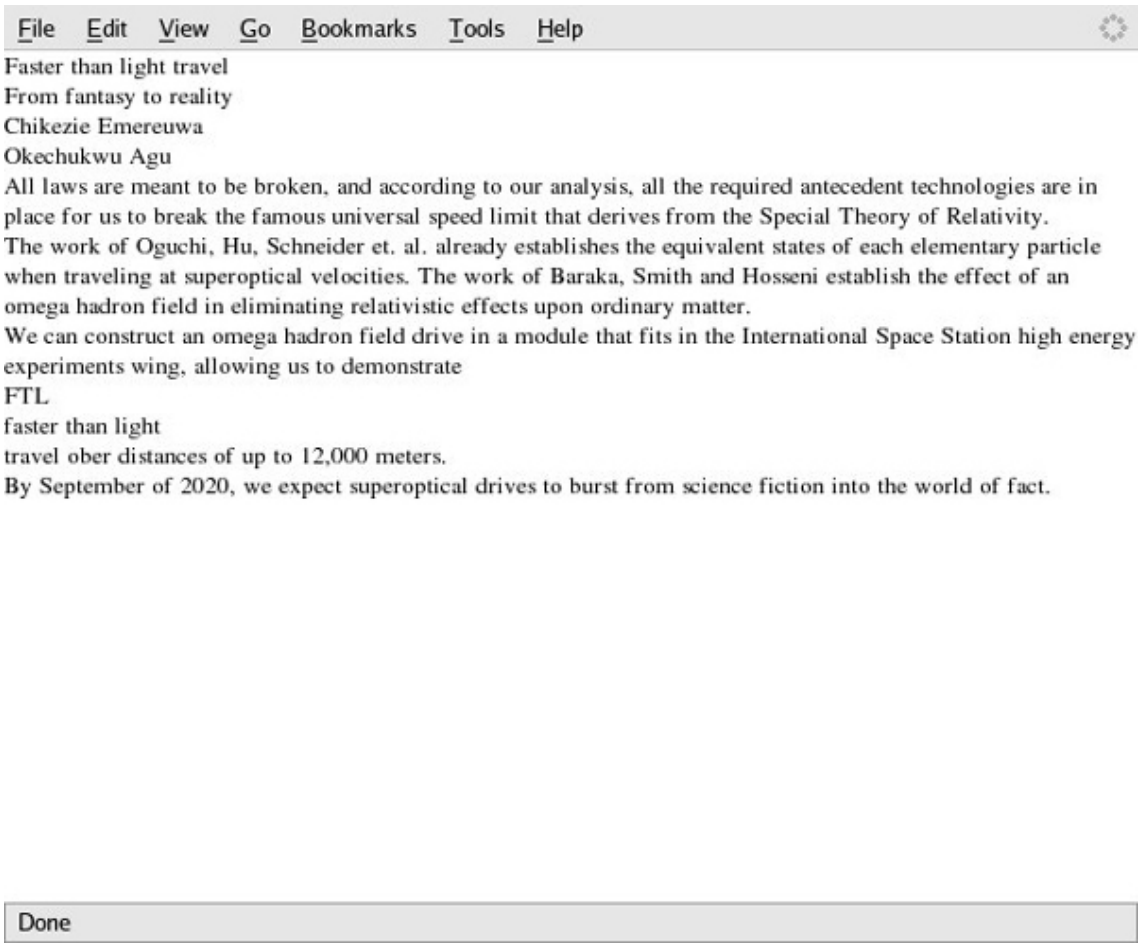
```
* {display: block;}
```

The * selector is new in CSS2 and selects any element.

You can apply this CSS to the sample XML in Default behavior displaying XML without any stylesheet on page 5 by adding the following processing instruction right below the XML declaration.

```
<?xml-stylesheet type="text/css" href="eg_3_1.css"?>
```

See eg_3_1.xml in the x-xmlcss-tutorial-files.zip for the full context. The above processing instruction follows the widely-implemented W3C specification for linking XML to stylesheets (whether in XSLT or CSS). As you can see from the resulting screen shot, now each element's content is in a separate line. This at least makes it a bit easier to follow, but you can do much better.

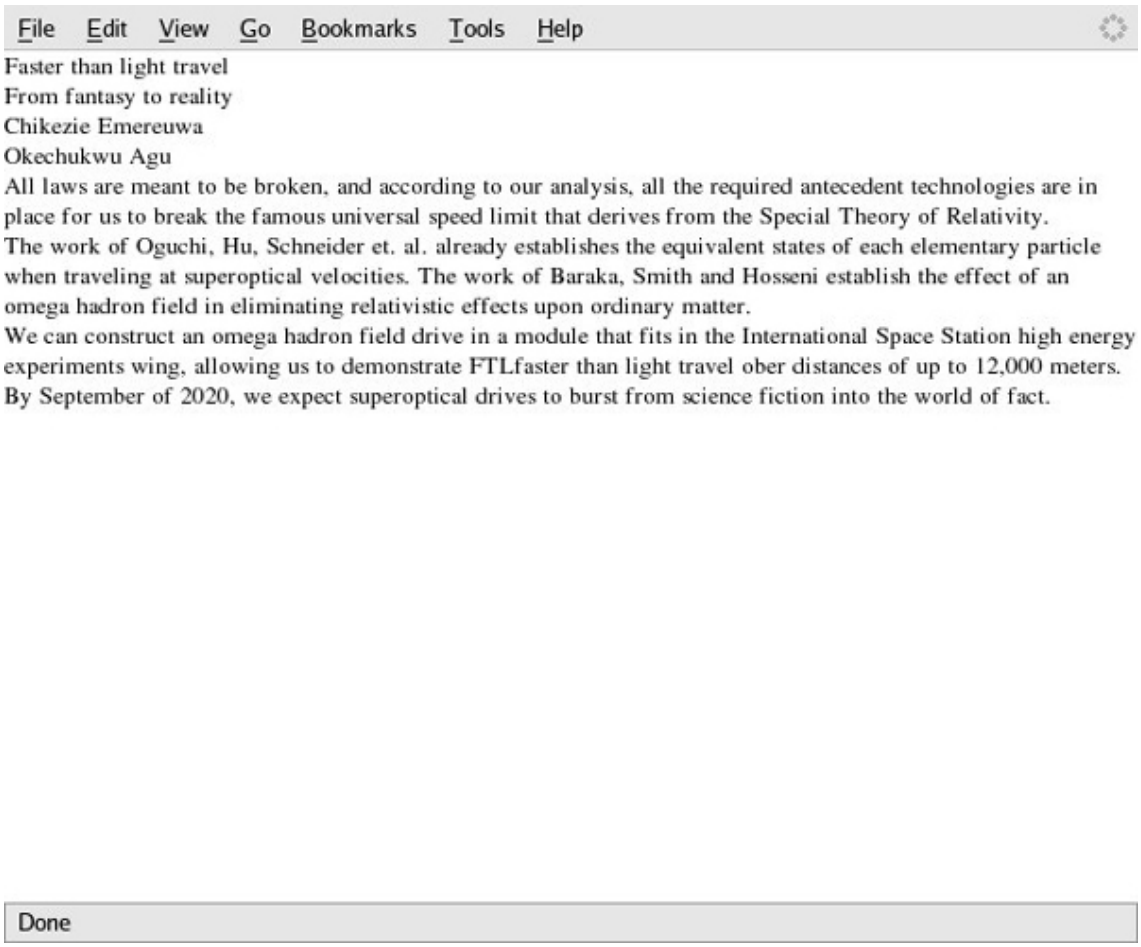


Display elements inline

Focusing on the `abbreviation` element from the sample document, it would be nice to have it flow inline into the paragraph in which it appears. The following CSS (eg_3_2.css in the x-xmlcss-tutorial-files.zip) achieves this:

```
abbreviation {display: inline;}

abbreviation * {display: inline;}

* {display: block;}
```

The `abbreviation *` selector matches all elements within `abbreviation`. The first and second rule can be combined using the selector `abbreviation, abbreviation *`. The result at least flows more nicely.



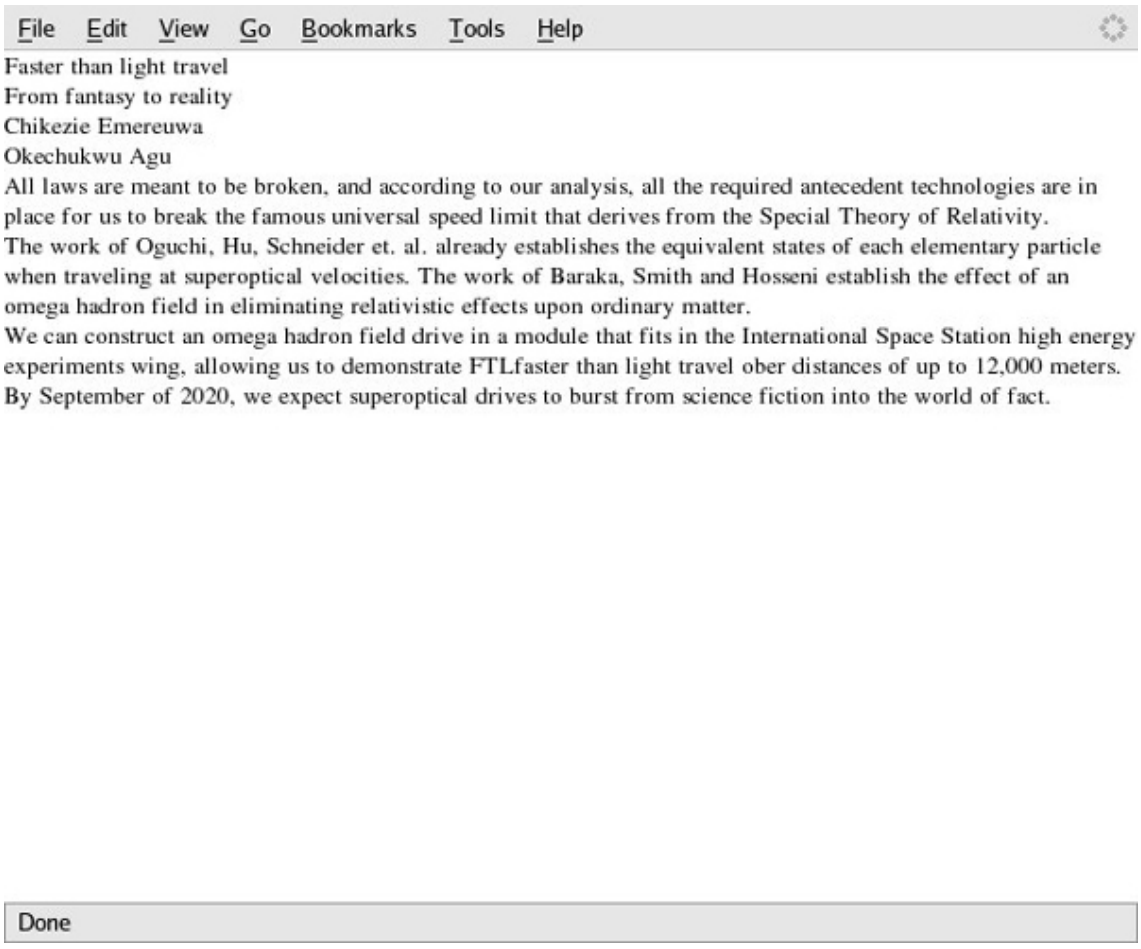
Inheriting the display of elements

If you're wondering why you have to specify the display for both the `abbreviation` element and its children, this is because the `display` property is not inherited (the browser does not refer to the parent of an element to determine the default value of `display`). If you only selected `abbreviation`, each child would fall back to the rule `* {display: block;}` and end up in its own block. Since `abbreviation` has no direct text content of its own, the result would be the same as in The display property on page 8. Similarly, if you only selected `abbreviation *` then `abbreviation` itself would fall under that catch-all rule and appear as a separate block.

CSS2 allows you to explicitly set the inheritance of `display` (or any property, for that matter), and this permits a more elegant way of expressing the presentation specified in Display elements inline on page 9. The following CSS is eg_3_3.css in the x-xmlcss-tutorial-files.zip.

```
abbreviation {display: inline;}

paper {display: block;}
```

```
* {display: inherit;}
```

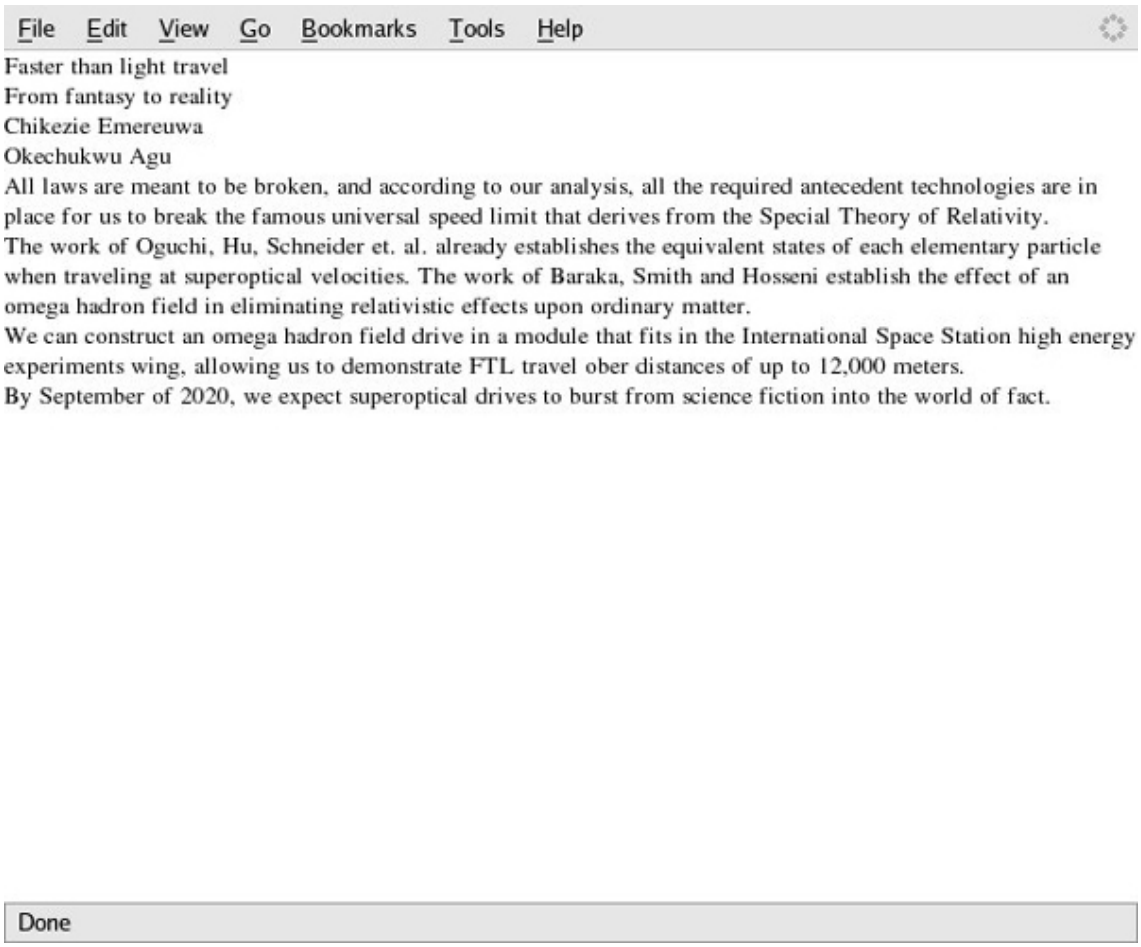


Suppress the display of elements

You have learned how to display XML elements inline, but in the example element `abbreviation` you probably want to show the abbreviation by default and its expansion. To completely suppress the display of an element, you use `display: none` as illustrated in the following CSS (eg_3_4.css in the x-xmlcss-tutorial-files.zip).

```
long {display: none;}

abbreviation {display: inline;}

paper {display: block;}

* {display: inherit;}
```

These four display property values are the foundation for using CSS to render XML in browsers.


File Edit View Go Bookmarks Tools Help
Faster than light travel
From fantasy to reality
Chikezie Emereuwa
Okechukwu Agu
All laws are meant to be broken, and according to our analysis, all the required antecedent technologies are in place for us to break the famous universal speed limit that derives from the Special Theory of Relativity.
The work of Oguchi, Hu, Schneider et. al. already establishes the equivalent states of each elementary particle when traveling at superoptical velocities. The work of Baraka, Smith and Hosseni establish the effect of an omega hadron field in eliminating relativistic effects upon ordinary matter.
We can construct an omega hadron field drive in a module that fits in the International Space Station high energy experiments wing, allowing us to demonstrate FTL travel ober distances of up to 12,000 meters.
By September of 2020, we expect superoptical drives to burst from science fiction into the world of fact.
Done

Section 4. Generate content

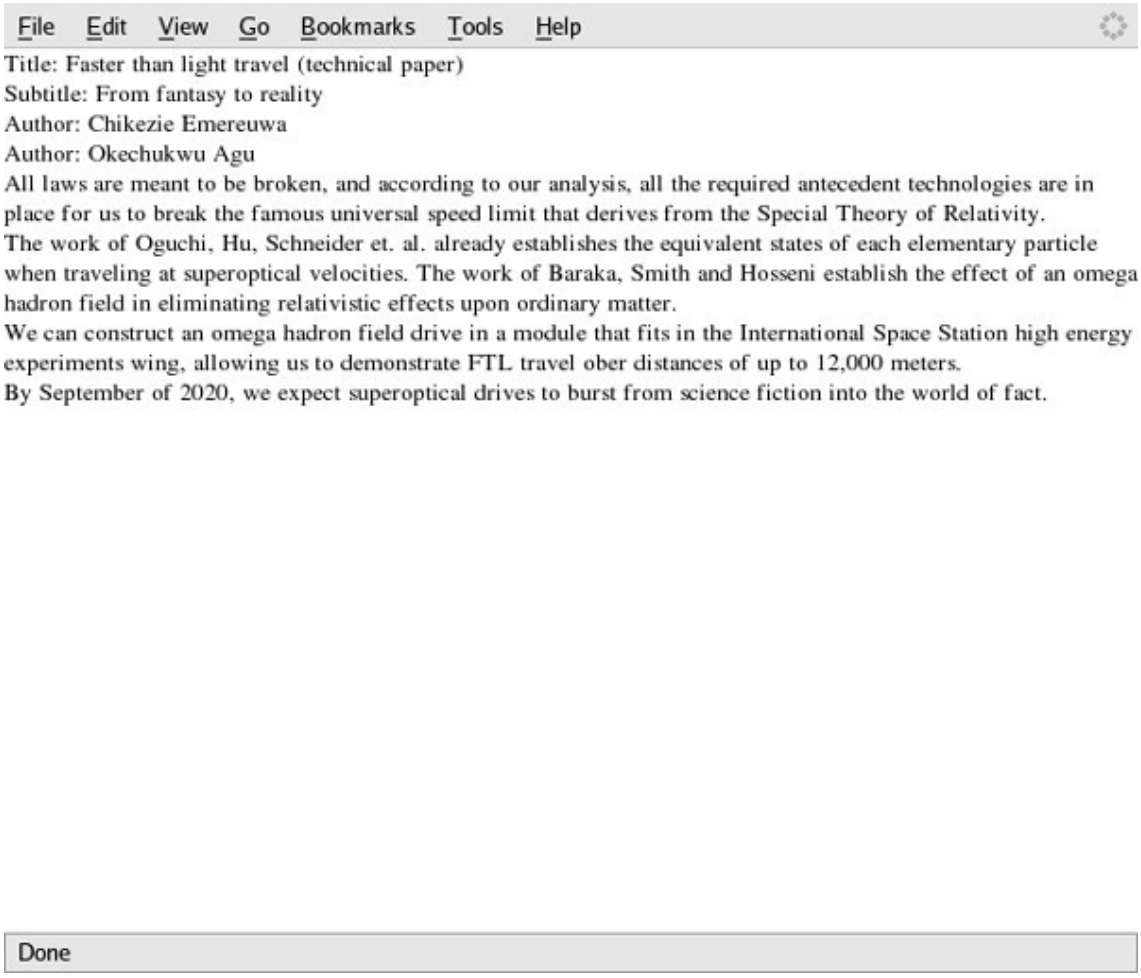
It's not always all in the source document

CSS2 introduced the ability to add text that is not present in the original content. This is not unlike text you might have in an XSLT stylesheet for copying to output, say within an `xml:text` instruction. In XSLT you pretty much have full control over added text (and even elements and other markup) for output. In CSS you're a bit more limited. In general, you can insert the content directly before or after an element that's specified by a selector by using the `:before` or `:after` specifier. You then use a property named `content` to provide the text to be inserted.

The following CSS (eg_4_1.css in the x-xmlcss-tutorial-files.zip) demonstrates generated content by adding captions to mark the title, subtitle, and authors of documents:

```
title:before {content: 'Title: ';}

title:after {content: ' (technical paper)';}

subtitle:before {content: 'Subtitle: ';}

author:before {content: 'Author: ';}

long {display: none;}

abbreviation {display: inline;}

paper {display: block;}

* {display: inherit;}
```

The CSS2 specification is not clear on whether or not you can generate elements in such content. But most observers agree that this is entirely contrary to the spirit of CSS, and certainly browsers that currently support generated content render embedded tags verbatim.



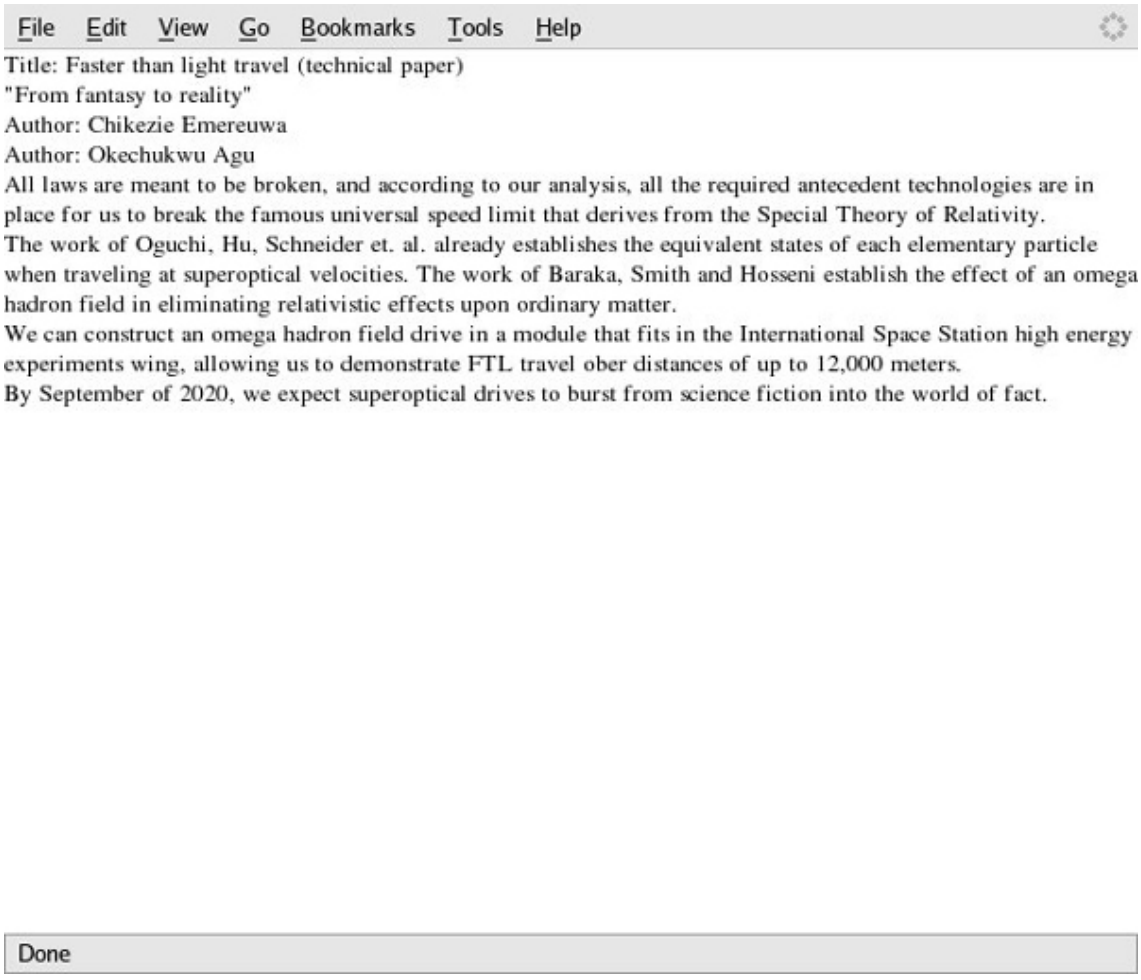
Generate quotation marks

Many people, especially programmers, don't think all that much about quotation marks. One gets used to the over-simplified model of quotes in ASCII and popular European character encodings, forgetting the typographical richness and flexibility traditionally associated with quotation marks. It has always been difficult to precisely control the characters used within multiple levels of nesting of quotations. To complicate things further, many different locales use different systems of quotation punctuation. CSS2's includes the ability to generate quotation marks with some intelligence, using the `open-quote` `close-quote` values for the `content` property.

The following CSS (eg_4_2.css in the x-xmlcss-tutorial-files.zip) uses quotes to punctuate subtitles:

```
title:before {content: 'Title: ';}

title:after {content: ' (technical paper)';}

subtitle:before {content: open-quote;}
```

```
subtitle:after {content: close-quote;}

author:before {content: 'Author: ';}

long {display: none;}

abbreviation {display: inline;}

paper {display: block;}

* {display: inherit;}
```



Special quote characters in generated content

HTML and XML have similar methods of escaping special characters, but CSS uses a different mechanism. For example, to unmistakably express a new line in XML or HTML you would use the character entity `
` (or `
`). In CSS, you would use the escape sequence `\A`.

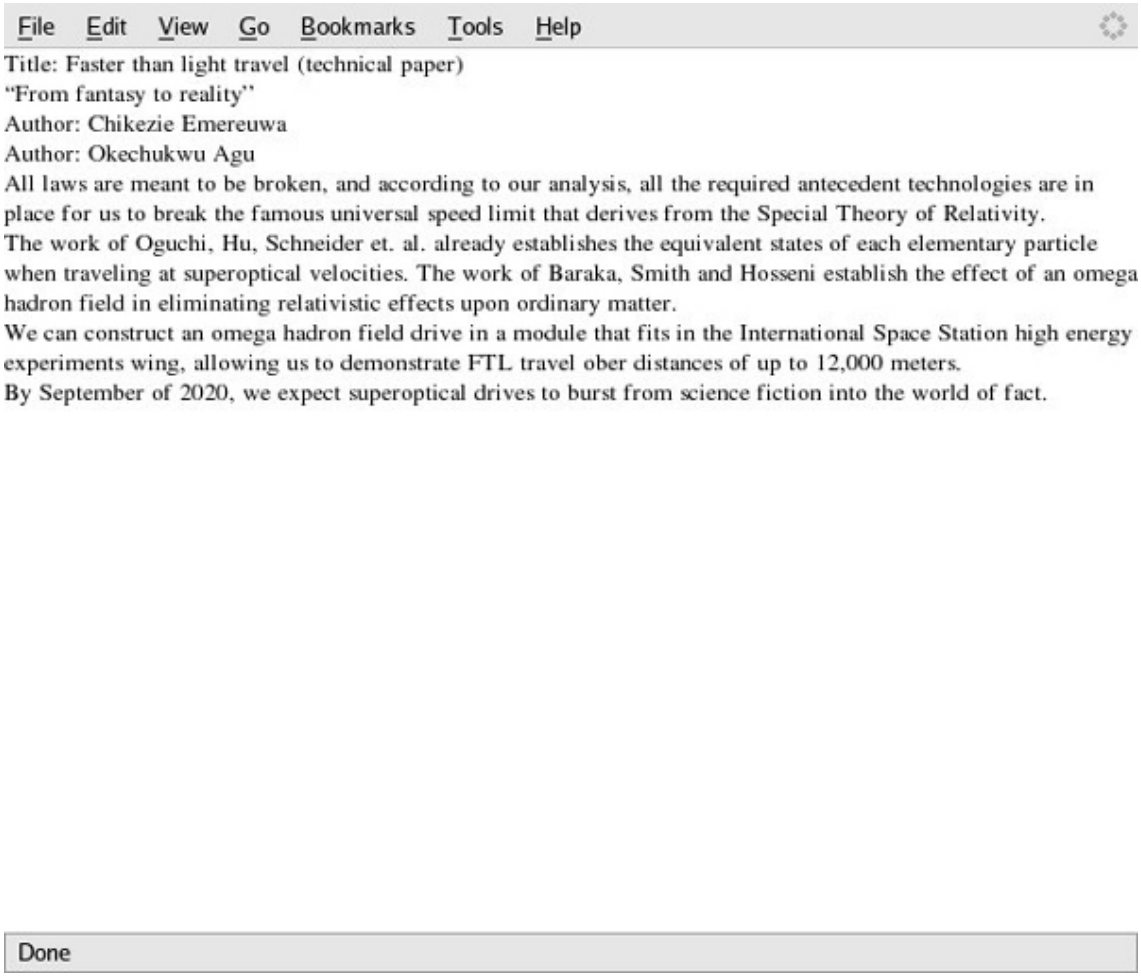
In the previous panel, Generate quotation marks on page 14, you learned how

to generate quotation marks. CSS2 also allows you to control what characters are used for the punctuation; the `quotes` property specifies the characters to be used at alternate levels of quote nesting.

The following CSS (eg_4_3.css in the x-xmlcss-tutorial-files.zip) uses left and right curly quote characters to punctuate subtitles:

```
title:before {content: 'Title: ';}

title:after {content: ' (technical paper)';}

* {quotes: '\201C' '\201D' '\2018' '\2019';}

subtitle:before {content: open-quote;}

subtitle:after {content: close-quote;}

author:before {content: 'Author: ';}

long {display: none;}

abbreviation {display: inline;}

paper {display: block;}

* {display: inherit;}
```

The four characters specified in `quotes` are to be used for left and right quotes at odd nesting levels, then even nesting levels. You can use any character you like as a quotation mark.



White space in generated content

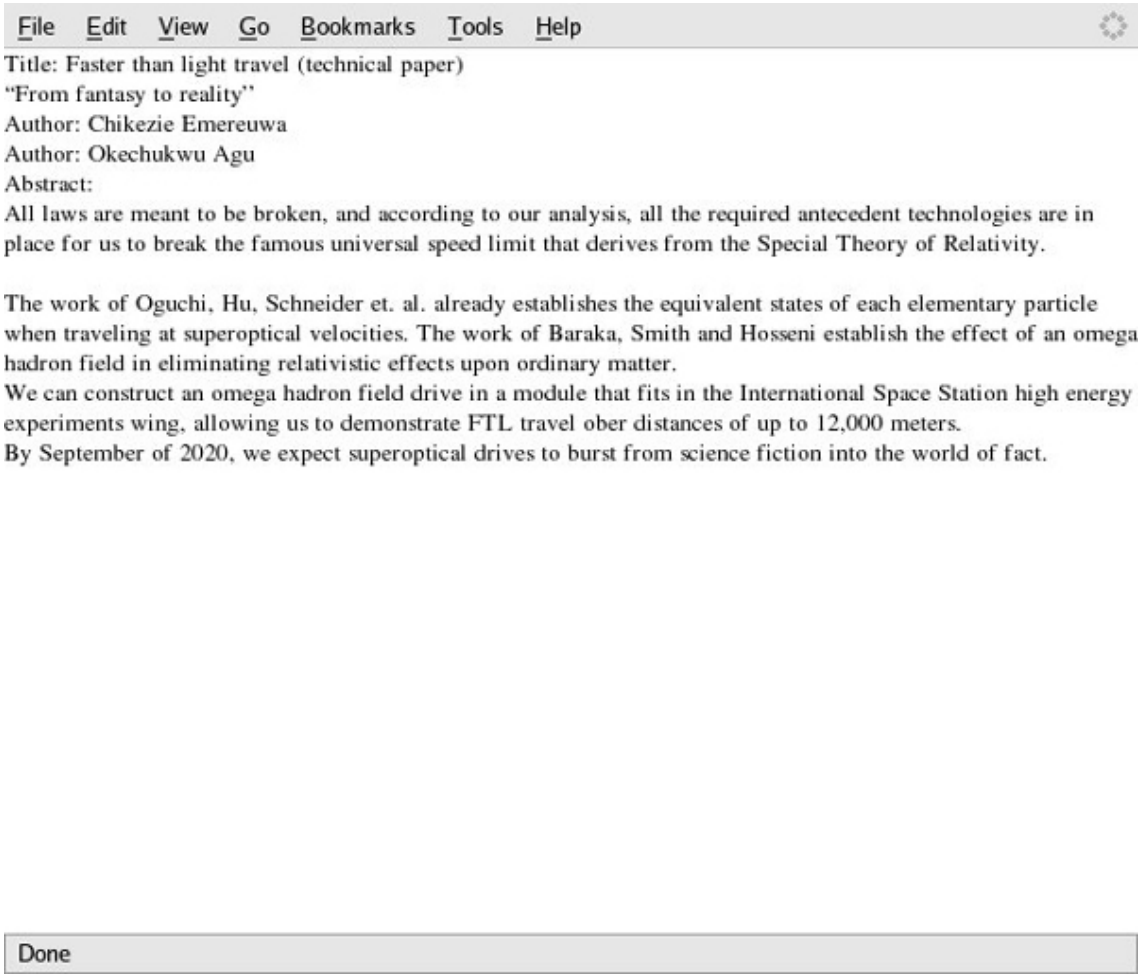
You might want to use white space in generated content. Usually if you just want to pad or space things out, you have a large array of CSS properties for the purpose (which I touch on later in this tutorial). But if you actually want inserted content to contain white space characters, be aware of a few things. First of all, white space in generated content follows the same display rules as white space in original content. You can use the `white-space` property to control this more finely.

The following CSS (eg_4_4.css in the x-xmlcss-tutorial-files.zip) specifies generated content as pre-formatted in order to preserve white space.

```
title:before {content: 'Title: ';}

title:after {content: ' (technical paper)';}

* {quotes: '\201C' '\201D' '\2018' '\2019';}

subtitle:before {content: open-quote;}
```

```
subtitle:after {content: close-quote;}

abstract:before {content: 'Abstract:\A'; white-space: pre;}

abstract:after {content: '\A\A0'; white-space: pre;}

author:before {content: 'Author: ';}

long {display: none;}

abbreviation {display: inline;}

paper {display: block;}

* {display: inherit;}
```

Notice the value `'\A\A0'`. For some reason, which I think might be a bug in Firefox, I had to add that non-breaking space (`\A0`, popularly known in HTML as ` `) for the preceding newline (`\A`) to take effect.



Handy notes on characters

You might at some point need to follow an escape sequence with alphanumeric characters. If so, you must be careful. For example, `\Aface` would be rendered as a single character rather than five. This is because escape sequences can be as many as six hexadecimal digits. If you want to be sure, pad the escape sequence with zeroes. For example `\00000Aface` is indeed the full five characters.

Speaking of characters, I mentioned that you can use any character you like as a quotation mark, but the following table lists the most common quotation mark characters (as noted in the CSS2 spec):

Character	CSS escape sequence	Unicode description
"	\0022	Quotation mark [ASCII double quotation mark]
'	\0027	Apostrophe [ASCII single quotation mark]
‹	\2039	Single left-pointing angle quotation mark
›	\203A	Single right-pointing angle quotation mark
«	\00AB	Left-pointing double angle quotation mark
»	\00BB	Right-pointing double angle quotation mark
‘	\2018	Left single quotation mark [single high-6]
’	\2019	Right single quotation mark [single high-9]
“	\201C	Left double quotation mark [double high-6]
”	\201D	Right double quotation mark [double high-9]
„	\201E	Double low-9 quotation mark [double low-9]

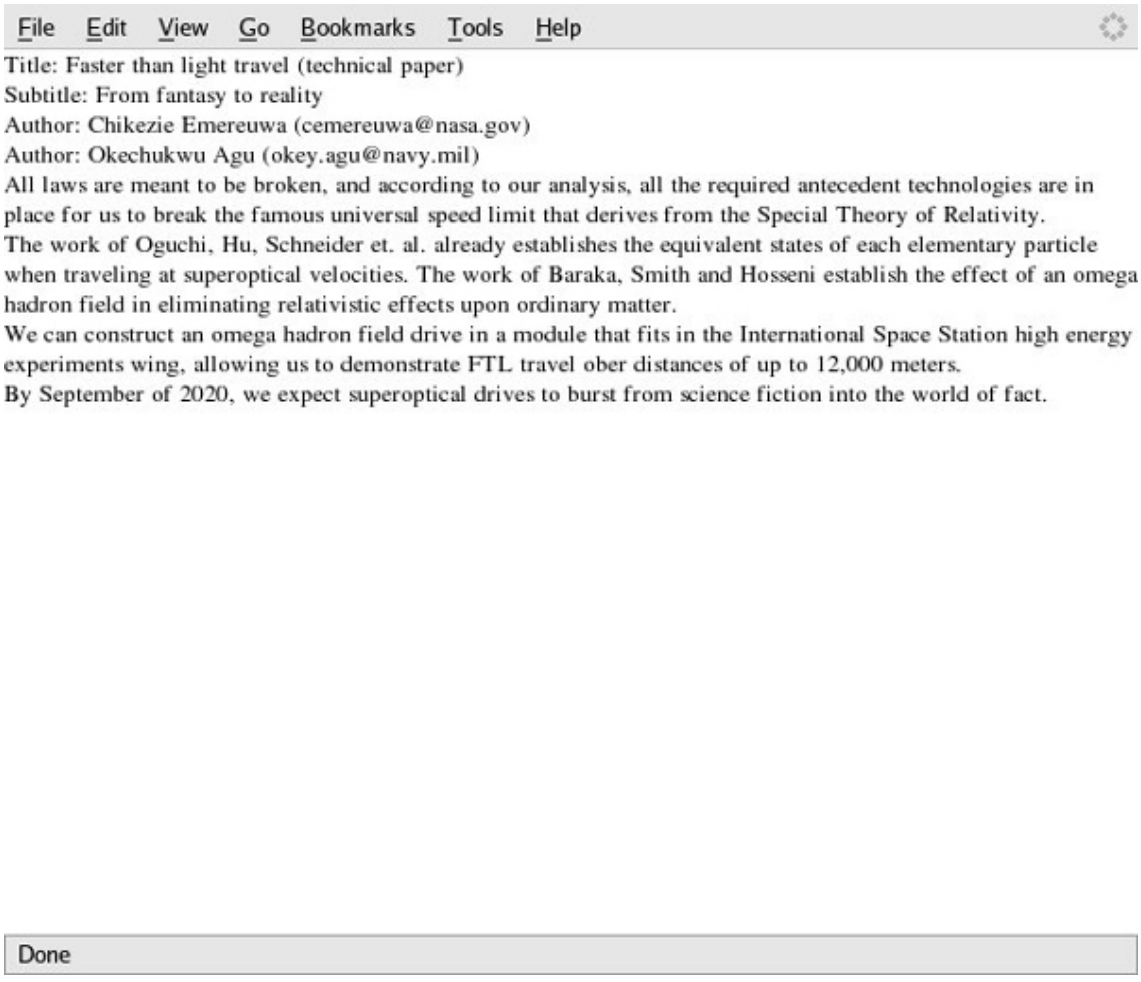
Working with attributes

CSS2 also provides support for working with XML (or HTML) attributes. One feature is the ability to refer to attribute values in generated content (and any other property value) using the `attr` function. The following CSS (eg_4_5.css in the x-xmlcss-tutorial-files.zip) takes advantage of this to display authors' e-mail addresses.

```
author:before {content: 'Author: ';}

author:after {content: ' (' attr(e-mail) ')';}

title:before {content: 'Title: ';}

title:after {content: ' (technical paper)';}

subtitle:before {content: 'Subtitle: ';}

long {display: none;}

abbreviation {display: inline;}
```

```
paper {display: block;}

* {display: inherit;}
```

Here is one area where you might miss the ability to generate tags in content. Standard CSS does not really offer any way to make the e-mail addresses proper links. To do so, you would need browser scripting or some such mechanism. (Opera does have some powerful, non-standard CSS extensions that you can use for creating links.)



CSS counters

It is often useful to number items that correspond to elements in the source XML. This is the reason for the sophisticated `xsl:number` instruction in XSLT. The closest equivalents in CSS are the `counter` and `counters` functions. The `counter-increment` and `counter-reset` properties are used to control counters. The following CSS (eg_4_6.css in the x-xmlcss-tutorial-files.zip) uses counters to mark the sections.

```
section:before {
  content: "Section " counter(section) "\A\A0";
  counter-increment: section;
  white-space: pre;
}

title:before {
  counter-reset: section;
  content: 'Title: ';
}

title:after {content: ' (technical paper)';}

subtitle:before {content: open-quote;}

subtitle:after {content: close-quote;}

author:before {content: 'Author: ';}

long {display: none;}

abbreviation {display: inline;}

paper {display: block;}

* {display: inherit;}
```

Use the `counter-reset` property to set a counter to a specific value. The default value, as in the example, is zero. If you use nested counters you will probably use this property to set inner counters back to zero once an outer counter has been updated.

Unfortunately, counters are not yet supported in Firefox so I cannot show a sample result image. Apparently, Opera is the only browser that does support CSS counters. Opera is a commercial browser that's well respected for its standards compliance, and you should consider it if you can (see Resources on page38).

Insert content from URIs

You do not have to embed generated content in the CSS. You can insert content that's read from a URL, as demonstrated in the following CSS (eg_4_7.css in the x-xmlcss-tutorial-files.zip), which uses left and right curly quote characters to punctuate subtitles:

```
paper:after {
  content: url('disclaimer.txt');
  white-space: pre;
}

title:before {content: 'Title: ';}

title:after {content: ' (technical paper)';}

* {quotes: '\201C' '\201D' '\2018' '\2019';}
```

```
subtitle:before {content: open-quote;}

subtitle:after {content: close-quote;}

author:before {content: 'Author: ';}

long {display: none;}

abbreviation {display: inline;}

paper {display: block;}

* {display: inherit;}
```

You can even insert images from external URLs if the media type indicates an image format supported by your browser. Unfortunately, content from URLs is not yet supported at all in Firefox, so I cannot show a sample result. Apparently, Opera is the only browser that does support the `url` function in the `content` property value.

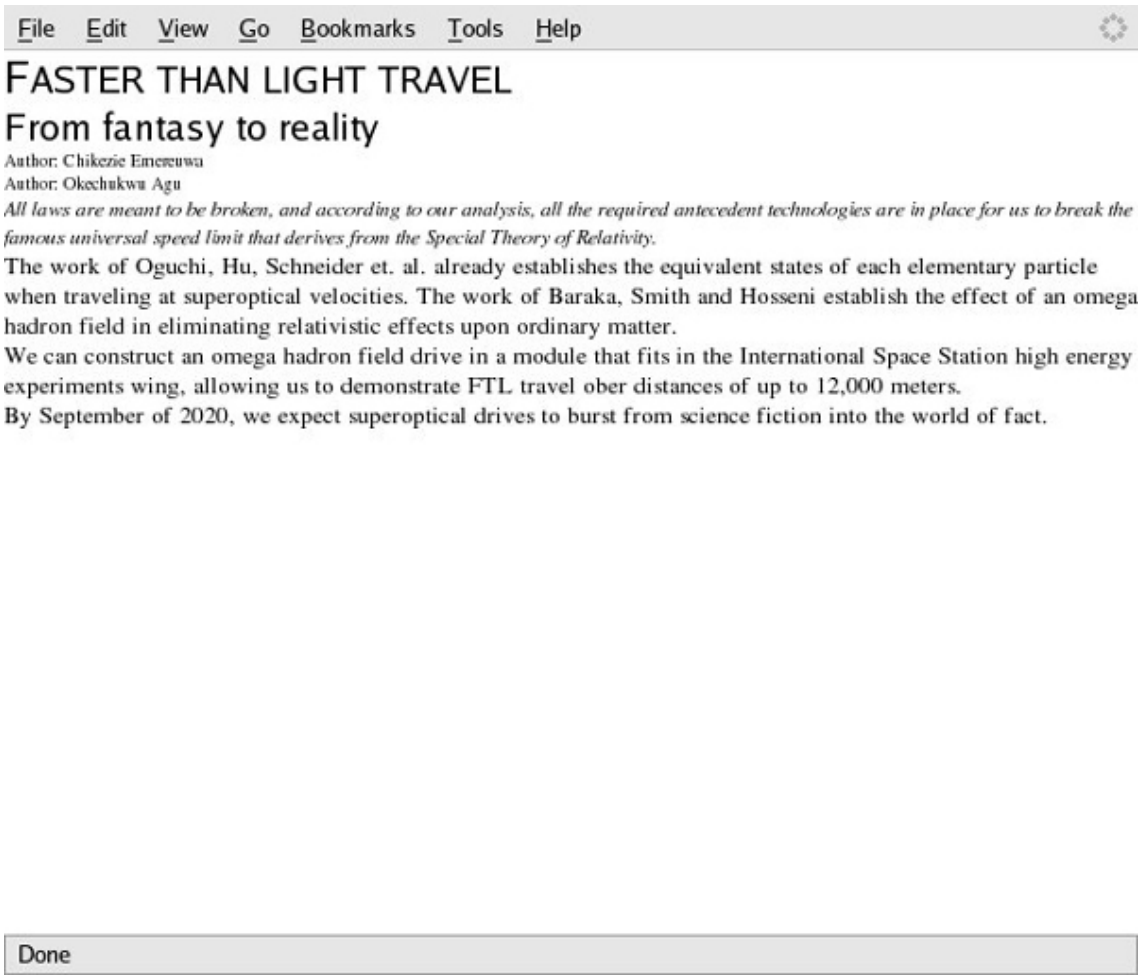
Section 5. Visual effects (and beyond)

Control type

Fine-tuning look and feel is the predominant usage of CSS. You can build on the basics learned so far to create all sorts of visual effects with XML and CSS. Much of this is material you are already familiar with from using CSS with HTML, but it's worth having some good examples of how to tweak appearance with the combination of XML-aware selectors and display properties.

The following CSS (eg_5_1.css in the x-xmlcss-tutorial-files.zip) uses typeface-related properties to render titles and subtitles more distinctly, and to format the abstract specially:

```
title {
  text-align: left;
  font: small-caps 175% sans-serif;
}

subtitle {
  text-align: left;
  font: 150% sans-serif;
}

abstract { font-size: small; font-style: italic;}

author:before {
  content: 'Author: ';
  font-size: x-small;
  font-weight: lighter;
}

author {
  font-size: x-small;
  font-weight: lighter;
}

long {display: none;}

abbreviation {display: inline;}

paper {display: block;}

* {display: inherit;}
```



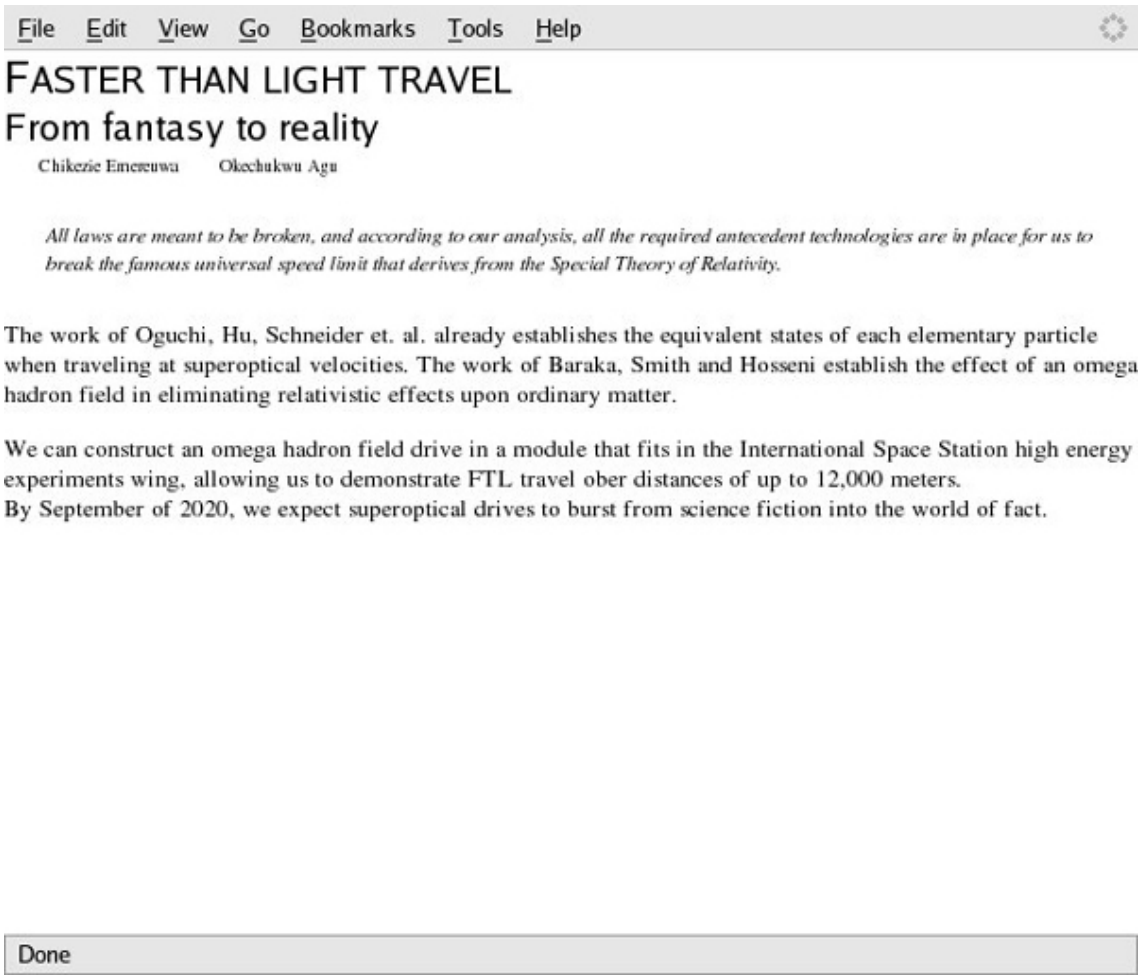
Control spacing

Judicious use of space can have a tremendously positive effect on the look of a page. All the CSS spacing properties are also available for use with XML, as demonstrated in the following listing (eg_5_2.css in the x-xmlcss-tutorial-files.zip):

```
title {
  text-align: left;
  font: small-caps 175% sans-serif;
}

subtitle {
  text-align: left;
  font: 150% sans-serif;
}

abstract {margin: 2em;}

abstract {font-size: small; font-style: italic;}

author {
```

```
    display: inline;
    font-size: x-small;
    font-weight: lighter;
    padding-left: 2em;
  }

  section {
    margin: 1em 0 1em 0;
  }

  long {display: none;}

  abbreviation {display: inline;}

  paper {display: block;}

  * {display: inherit;}
```

The example specifies spacing in several ways, using variations on the `margin` property. This property controls spacing outside the (possibly invisible) border and the `padding` property, which controls spacing within the border. Syntax such as `margin: 1em` controls spacing on all four sides of the box. Syntax such as `margin-left: 1em` controls spacing on the specified side. Syntax such as `margin: 1em 2em 3em 4em` (known as groupings) controls spacing separately on each side, in the order top, right, bottom, left. I remember this by imagining I'm reading a map and that the four cardinal points (clockwise) North, East, South, and West ("Never Eat Soggy Waffles" in my wife's favorite mnemonic) correspond to the order of directions given in the grouping.



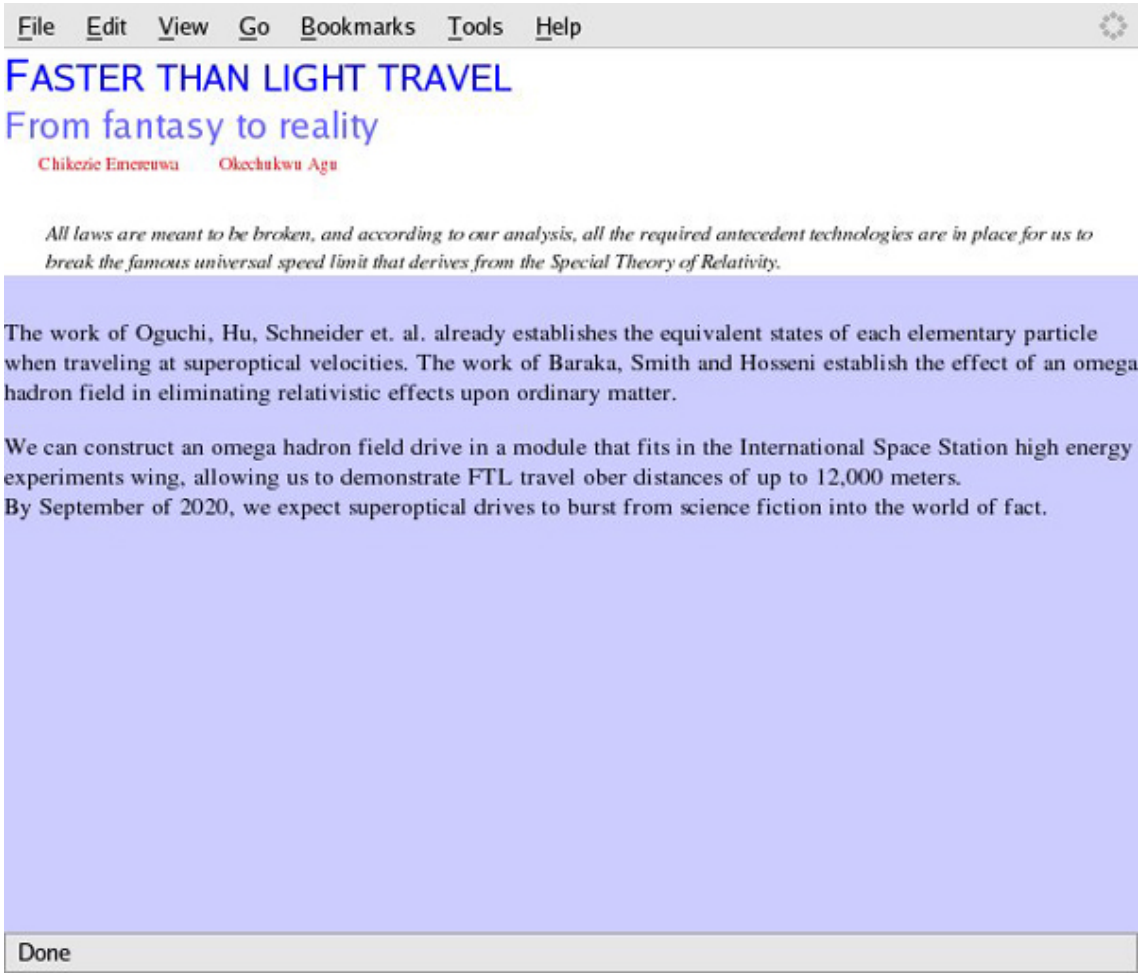
Control color

You can also control color and backgrounds for XML content as demonstrated in the following CSS (eg_5_3.css in the x-xmlcss-tutorial-files.zip):

```
paper {
  background: #ccccff;
}

prologue {
  background: white;
}

title {
  color: blue;
  text-align: left;
  font: small-caps 175% sans-serif;
}

subtitle {
  color: #6666ff;
  text-align: left;
```

```
  font: 150% sans-serif;
}

abstract {margin: 2em;}

abstract {font-size: small; font-style: italic;}

author {
  display: inline;
  color: red;
  font-size: x-small;
  font-weight: lighter;
  padding-left: 2em;
}

section {
  margin: 1em 0 1em 0;
}

long {display: none;}

abbreviation {display: inline;}

paper {display: block;}

* {display: inherit;}
```

The example demonstrates text and background color properties specified using both named colors (such as `blue`) and RGB form (such as `#ccccff`).



Aural properties

As the CSS2 spec states: "The aural rendering of a document, already commonly used by the blind and print-impaired communities, combines speech synthesis and 'auditory icons.'" It is always a good idea to at least think of how you might present the content through a speech interface. If nothing else, consider that you may not want the content of some elements to be read. The following CSS (eg_5_4.css in the x-xmlcss-tutorial-files.zip) adds an aural style sheet section:

```
@media aural {
  /* Aural properties */
  title { pause: 5ms 20ms; richness: 80; }

  section { pause: 20ms; }

  abbreviation long {speak: none;}
}

paper {
  background: #ccccff;
```

```
}

prologue {
  background: white;
}

title {
  color: blue;
  text-align: left;
  font: small-caps 175% sans-serif;
}

subtitle {
  color: #6666ff;
  text-align: left;
  font: 150% sans-serif;
}

abstract {margin: 2em;}

abstract {font-size: small; font-style: italic;}

author {
  display: inline;
  color: red;
  font-size: x-small;
  font-weight: lighter;
  padding-left: 2em;
}

section {
  margin: 1em 0 1em 0;
}

long {display: none;}

abbreviation {display: inline;}

paper {display: block;}

* {display: inherit;}
```

Section 6. Lists and tables

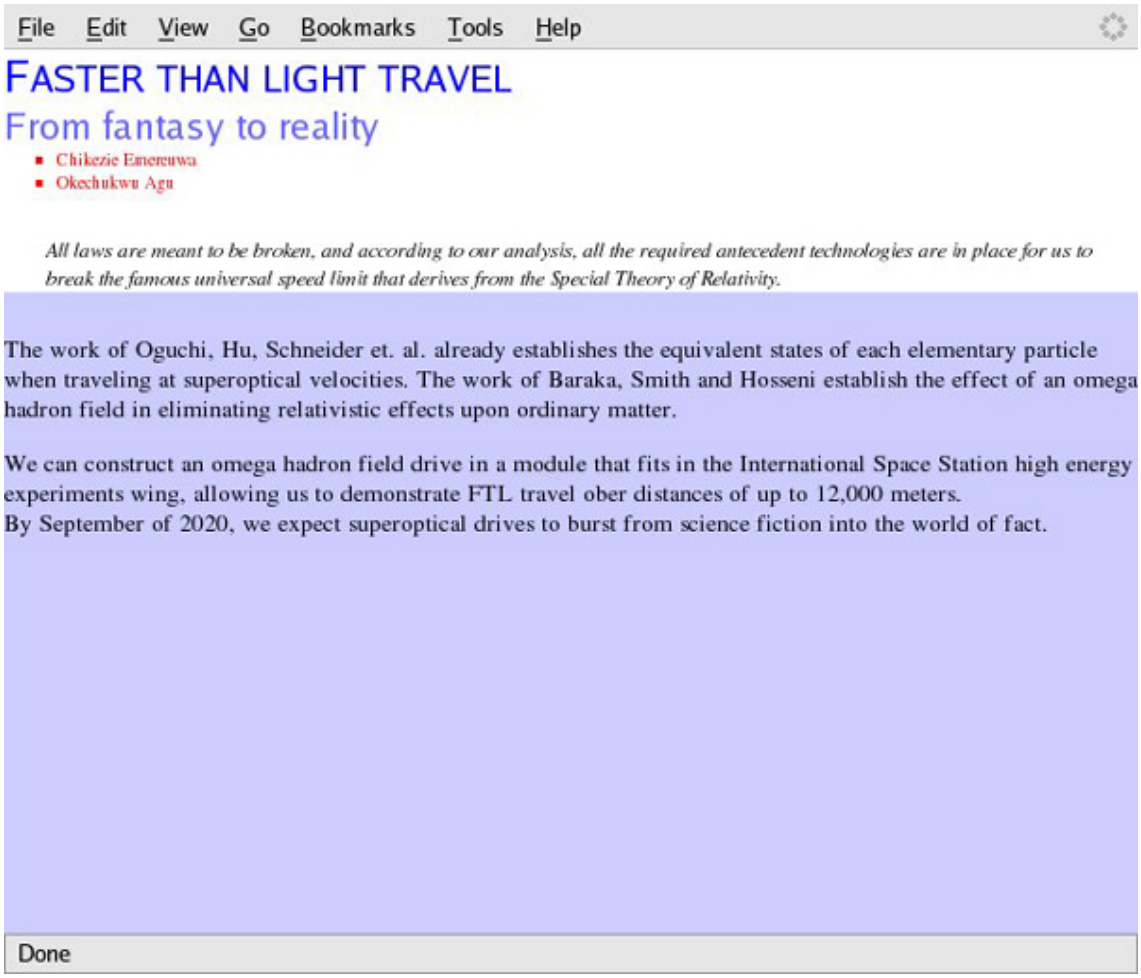
Display lists

CSS2 provides `display` properties for formatting lists and tables. You have a huge amount of flexibility in this, and the following CSS (eg_6_1.css in the x-xmlcss-tutorial-files.zip) just scratches the surface of how you might treat XML elements as lists for display. In this case, the `author` elements are rendered as list items.

```
author {
  display: list-item;
  list-style: square outside;
  margin-left: 3em;
  color: red;
  font-size: x-small;
  font-weight: lighter;
}

paper {
  background: #ccccff;
}

prologue {
  background: white;
}

title {
  color: blue;
  text-align: left;
  font: small-caps 175% sans-serif;
}

subtitle {
  color: #6666ff;
  text-align: left;
  font: 150% sans-serif;
}

abstract {margin: 2em;}

abstract {font-size: small; font-style: italic;}

section {
  margin: 1em 0 1em 0;
}

long {display: none;}

abbreviation {display: inline;}

paper {display: block;}

* {display: inherit;}
```

The key is the rule `display: list-item`, which opens up all the available list styles and decorations. I choose to use a square glyph as the list bullet item,

and to place it outside the bounding box for the display item. Be careful if you do this -- you need to leave room for any list markings that would be placed outside the bounding box. For example, if you delete the `margin-left: 3em` rule, this pushes the author text to the left margin of the browser, leaving no room for the bullet items outside the bounding box, and they just disappear.



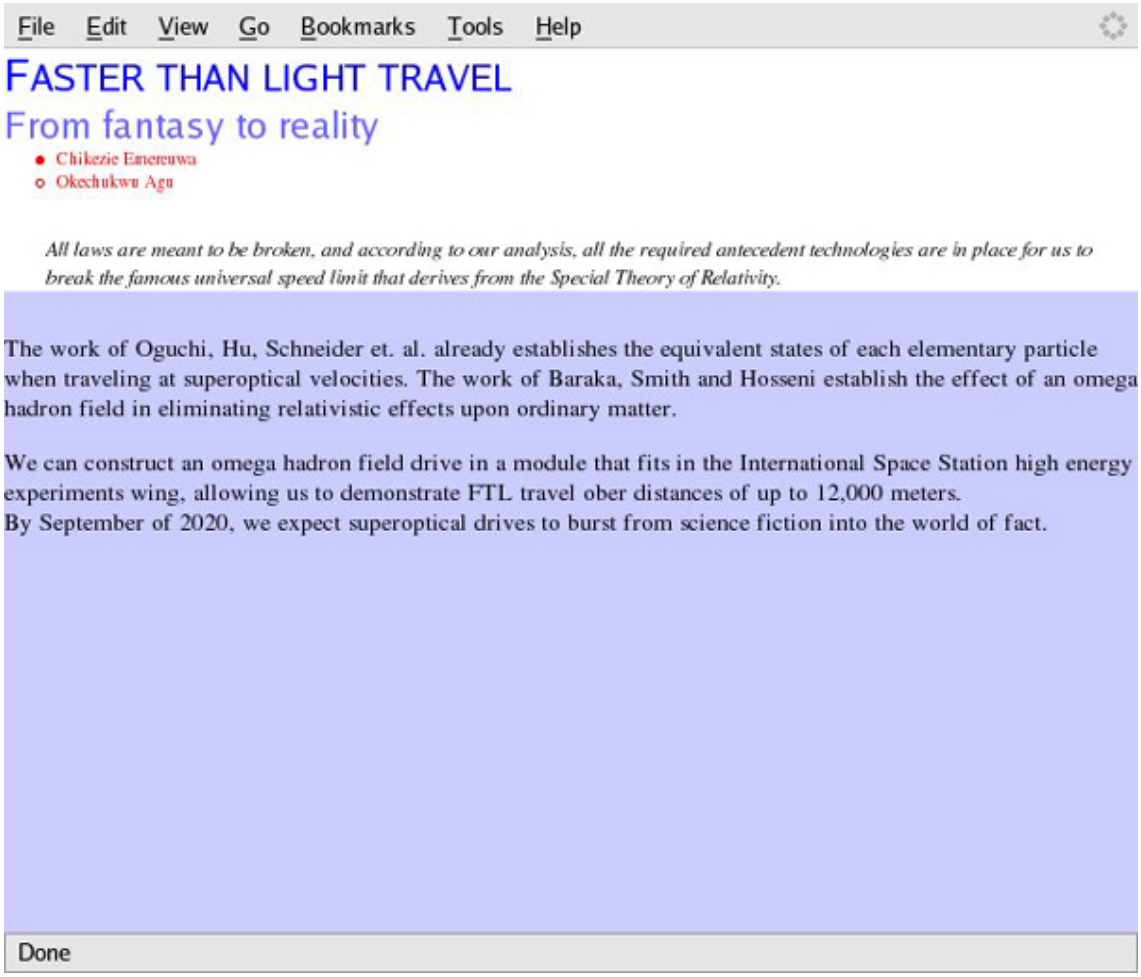
More fun with lists

You can do a lot with lists by just tweaking the style settings, but in order to have some real fun with the current example, you might want to differentiate list items according to the attributes on the source element. The following CSS (eg_6_2.css in the x-xmlcss-tutorial-files.zip) differentiates authors by using a filled disc as a bullet for authors marked as members, and unfilled circles for authors marked as non-members. It uses a special type of CSS2 selector for checking attribute values.

```
author[member='yes'] {
  list-style-type: disc;
}
```

```
author {
  display: list-item;
  list-style: circle outside;
  margin-left: 3em;
  color: red;
  font-size: x-small;
  font-weight: lighter;
}

paper {
  background: #ccccff;
}

prologue {
  background: white;
}

title {
  color: blue;
  text-align: left;
  font: small-caps 175% sans-serif;
}

subtitle {
  color: #6666ff;
  text-align: left;
  font: 150% sans-serif;
}

abstract {margin: 2em;}

abstract {font-size: small; font-style: italic;}

section {
  margin: 1em 0 1em 0;
}

long {display: none;}

abbreviation {display: inline;}

paper {display: block;}

* {display: inherit;}
```

The selector `author[member='yes']` captures elements with the given attribute value. It's much like the XSLT pattern `author[@member='yes']`. You probably remember that the *cascade* in CSS refers to how conflicting rules are resolved. The `author` element is caught by both the plain `author` selector and the more specific `author[member='yes']`. In fact, it is even caught by the * selector at the bottom of the CSS. In most cases, your browser supplies even more default rules. In CSS, the most specific rule generally prevails. So for authors with the specified attribute value, `list-style-type: disc` partially overrules the similar rule `list-style: circle outside` associated with the less specific selector. I say partially because `list-style: circle outside` is actually short for `list-style-type: circle; list-style-position: outside`, and only the *type* part is actually overruled in the cascade.



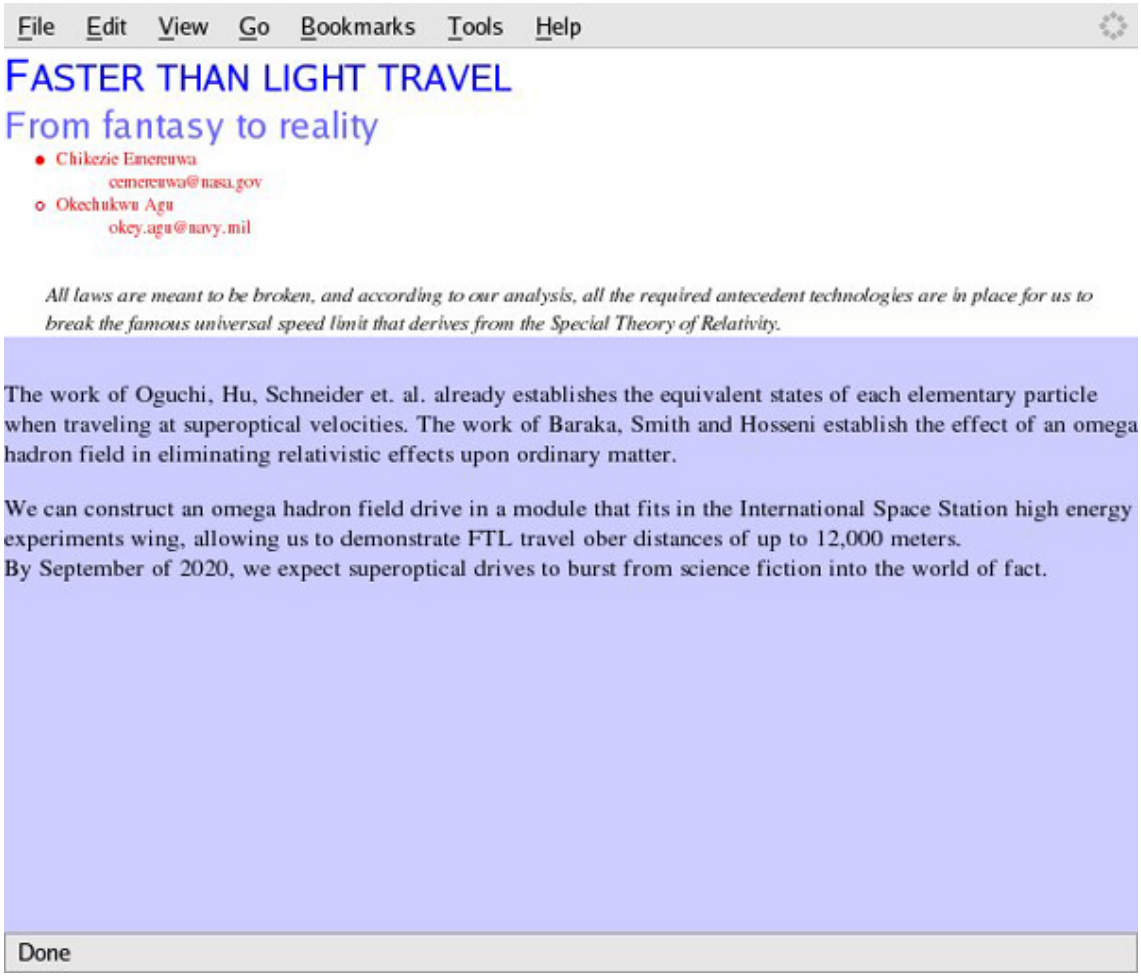
Mix generated content into lists

All the various content generation tricks are available within list items. It's especially important to remember that content generated within the `:before` and `:after` pseudo-elements is rendered within the bounding box of the actual element. This means that if you use generated content where the actual element has `display: list-item`, you're free to specify the generated item display also as `display: list-item` or even `display: block`; however, it is still rendered by the browser as part of the actual element. To illustrate this, the following CSS (eg_6_3.css in the x-xmlcss-tutorial-files.zip) renders an author e-mail address as a nested list item.

```
author[member='yes'] {
  list-style-type: disc;
}

author {
  display: list-item;
  list-style: circle outside;
  margin-left: 3em;
  color: red;
```

```
    font-size: x-small;
    font-weight: lighter;
  }

  author:after {
    display: list-item;
    list-style: none;
    content: attr(e-mail);
    margin-left: 3em;
    color: red;
    font-size: x-small;
    font-weight: lighter;
  }

  paper {
    background: #ccccff;
  }

  prologue {
    background: white;
  }

  title {
    color: blue;
    text-align: left;
    font: small-caps 175% sans-serif;
  }

  subtitle {
    color: #6666ff;
    text-align: left;
    font: 150% sans-serif;
  }

  abstract {margin: 2em;}

  abstract {font-size: small; font-style: italic;}

  section {
    margin: 1em 0 1em 0;
  }

  long {display: none;}

  abbreviation {display: inline;}

  paper {display: block;}

  * {display: inherit;}
```

Notice the rules `author { margin-left: 3em; }` and `author:after { margin-left: 3em; }`. You can see that both the author name and the e-mail address are rendered as list items with the same stated indentation; the indentation of the latter is set with respect to that of the former.



Display tables

CSS2 has just as rich a set of primitives for rendering tables. To better show how tables can be derived from XML structure, I use a different source document for this example. The following is a listing of papers recently published by The Tech Society.

```
<?xml version='1.0' encoding='utf-8'?>
<?xml-stylesheet type="text/css" href="eg_6_4.css"?>
<paper-listing>
  <paper>
    <title>Faster than light travel</title>
    <main-contact e-mail="cemereuwa@nasa.gov">Chikezie Emereuwa</main-contact>
    <publication-date>2004-11-02</publication-date>
  </paper>
  <paper>
    <title>Unlimited matter transmutation</title>
    <main-contact e-mail="ehud.dalan@techlab.il">Ehud Dalan</main-contact>
    <publication-date>2004-10-22</publication-date>
  </paper>
  <paper>
    <title>Automated language translation</title>
```

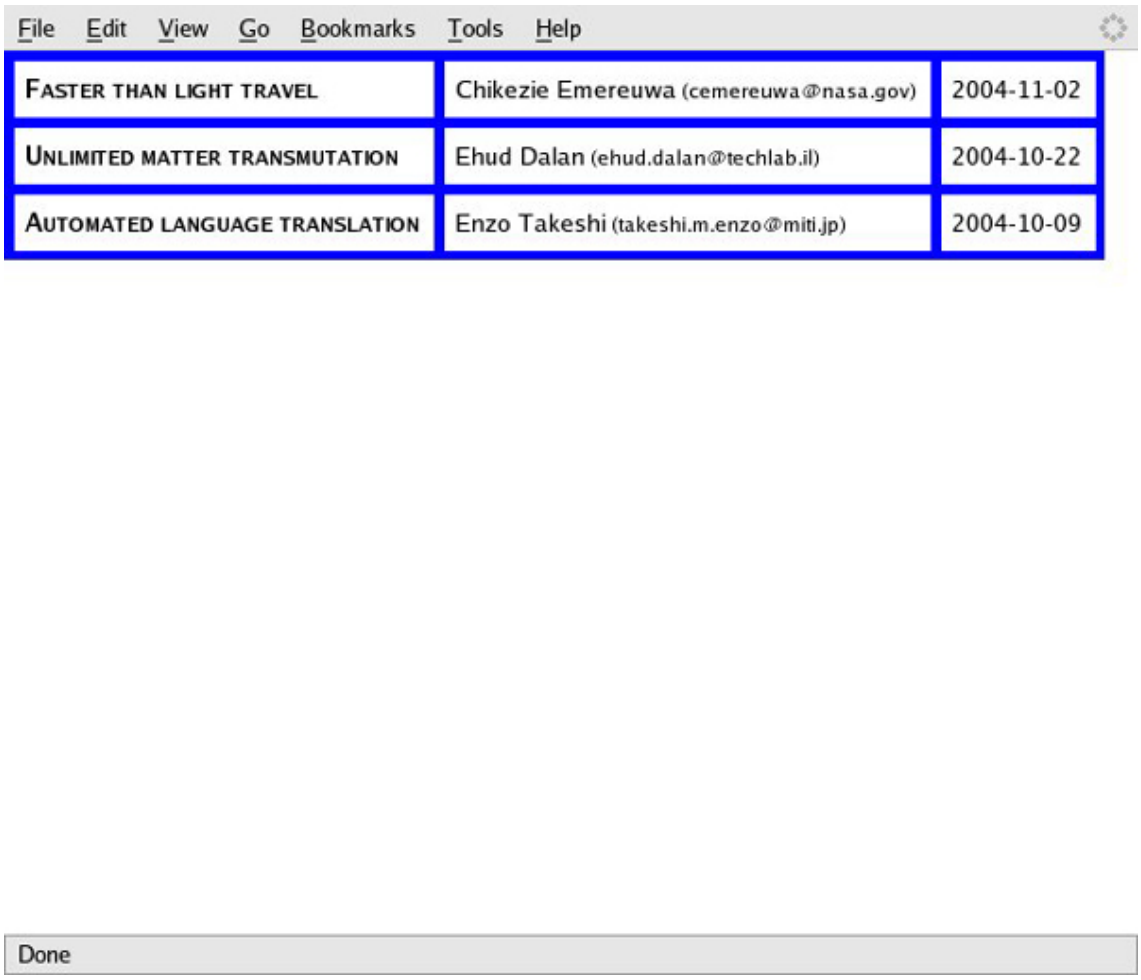
```
    <main-contact e-mail="takeshi.m.enzo@miti.jp">Enzo Takeshi</main-contact>
    <publication-date>2004-10-09</publication-date>
  </paper>
</paper-listing>
```

The following CSS (eg_6_4.css in the x-xmlcss-tutorial-files.zip) renders a corresponding table of articles in the browser. Again it barely scratches the surface of table display capabilities.

```
paper-listing {
  display: table;
  border: solid 2pt blue;
  font-family: sans-serif;
}

paper { display: table-row; }

paper * { padding: 0.5em; border: solid 2pt blue; }

title {
  display: table-cell;
  font-variant: small-caps;
  font-weight: bold;
}

main-contact {
  display: table-cell;
}

main-contact:after { content: ' (' attr(e-mail) ')'; font-size: smaller; }

publication-date { display: table-cell; }
```

You have to be careful about which elements own aspects of borders, spacing, and other style within the table. Unlike plain HTML, where you would do things like add `border=1 cellpadding=2` attributes in order to control these features of the whole table, you generally control them separately at the table, header, footer, body, row, and cell level in CSS.

You don't have to have an element for each bit of table structure (table, header, footer, body, row, and cell) to use CSS table display. CSS has a whole series of rules for what to do if, say, a table cell does not appear within a defined table row. It handles these by automatically creating **anonymous table objects** to effectively complete the structure behind the scenes.


File Edit View Go Bookmarks Tools Help
FASTER THAN LIGHT TRAVEL
Chikezie Emereuwa (cemereuwa@nasa.gov)
2004-11-02
UNLIMITED MATTER TRANSMUTATION
Ehud Dalan (ehud.dalan@techlab.il)
2004-10-22
AUTOMATED LANGUAGE TRANSLATION
Enzo Takeshi (takeshi.m.enzo@miti.jp)
2004-10-09
Done

Section 7. Wrap up

Summary

In this tutorial, you have learned how to use CSS to style XML. This included:

- Establishing the basic browser display
- Controlling font details and spacing
- Formatting lists and tables
- Inserting content that is not already present in the XML source
- The basics of how to determine style for speech reader browsers

This is just the beginning. By now, you probably have an idea of how versatile CSS is just from the examples in this tutorial. The only way to truly master the subject is through a lot of practice and experimentation.

One note of caution: CSS gives you a lot of gee-whiz display power, but just as I'd advise in styling HTML, don't forget the substance. Design for accessibility and the principle of *least surprise*. All the aural and other accessibility features for CSS are available, and I covered some of this in Aural properties on page 28. Don't neglect these considerations in your eagerness to position and size everything to pixel perfection.

Resources

You might find these resources helpful:

- If you're unfamiliar with XML, start with the many helpful resources in "*New to XML* (http://www-128.ibm.com/developerworks/xml/newto/) ."
- Learn the basics of CSS. Many start with CSS for HTML. You can find many articles on this topic in the developerWorks *Web architecture zone* (http://www.ibm.com/developerworks/web) , including:
  - "*Intro to cascading style sheets: Type*" by Dave Taylor (September 2001)
  - "*Tip: Using CSS2 to display XML documents*" by David Mertz (December 2001)
- Also see "*Adding a touch of style* (http://www.w3.org/MarkUp/Guide/Style) ", a great introduction to CSS by Dave Raggett of the W3C.
- Bookmark and refer to the formal CSS2 specification, "*Cascading Style Sheets, level 2: CSS2 Specification* (http://www.w3.org/TR/CSS2/) ." Many parts of the specification are clear and readable.
- The author tested the CSS in this tutorial with *Firefox* (http://www.getfirefox.com/) , a popular and free Web browser available on Windows, Mac OS X, Linux, and other platforms. Firefox is based on Mozilla's rendering engine, which has always been known to be a very

CSS-compliant browser. Also consider *Opera* (http://www.opera.com/) , a commercial (though inexpensive) browser implementing even more of CSS2 than Firefox.

- Use the *W3C CSS Validation Service* (http://jigsaw.w3.org/css-validator/) , "a free service that checks Cascading Style Sheets (CSS) in (X)HTML documents or standalone for conformance to W3C recommendations."
- Keep an eye on developments in *CSS3*, the next version of CSS. Many browsers already experimentally implement features proposed on CSS3 working drafts. Read "*CSS 3 Selectors*" by Russell Dyer for a good overview of the selectors (though CSS3 includes more than new selectors). But before CSS3 is complete, you can expect to see *CSS 2.1* (http://www.w3.org/TR/CSS21/) , a minor update.
- To learn more about the basics of using CSS with HTML, read "*Starting with HTML + CSS* (http://www.w3.org/Style/Examples/011/firstcss) " by Bert Bos of the W3C.
- If you're not already familiar with XPath, take one of several developerWorks tutorials on the topic:
  - "*Get started with XPath*" (May 2004), by Bertrand Portier
  - "*Create multi-purpose Web content with XSLT*" (March 2003)
- Browse a wide range of XML-related titles at the developerWorks *Developer Bookstore* (http://devworks.krcinfo.com/) , including books on *CSS*.
- Find more XML resources on the *developerWorks XML zone* (http://www.ibm.com/developerworks/xml/) .
- Finally, find out how you can become an *IBM Certified Developer in XML and related technologies* (http://www-1.ibm.com/certify/certs/adcdxmlrt.shtml) .

Feedback

Please let us know what you think of this tutorial. We look forward to hearing from you! Additionally, you are welcome to contact the author, Uche Ogbuji, at *uche.ogbuji@fourthought.com*.

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The open source Toot-O-Matic tool is an XSLT stylesheet and several XSLT extension functions that convert an XML file into a number of HTML pages, a zip file, JPEG heading graphics, and two PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML. (It also saves our production team a great deal of time and effort.)

For more information about the Toot-O-Matic, visit www-106.ibm.com/developerworks/xml/library/x-toot/ .