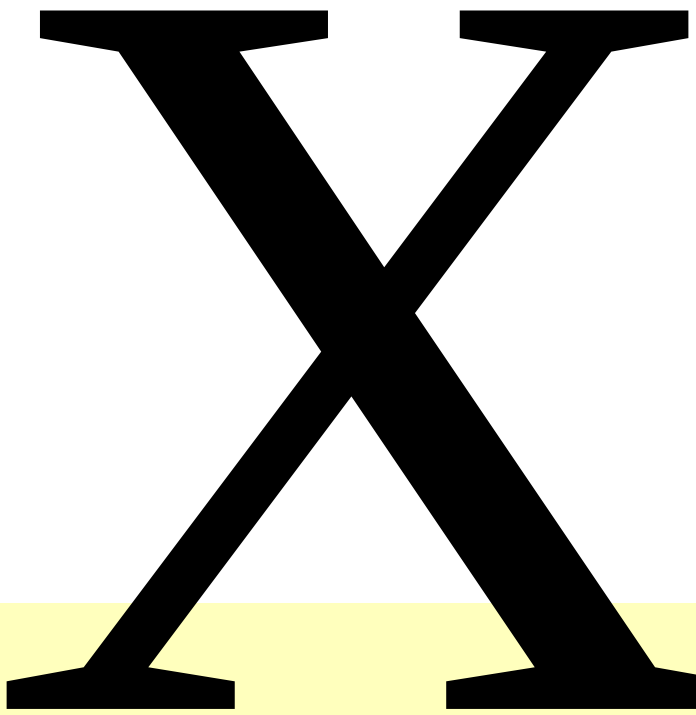
Customized Authoring
with Extensible Markup



XML 1.0

a Tutorial for Emilé 1.0

by Media Design in•Progress

This edition was completed April 4, 1999.

Lesson 1: Authoring Extensible HTML

The W3C has announced that the next version of HTML will be based on XML[1]. **Extensible HTML allows use of features from XML** in HTML documents.

XML and Extensible HTML is supported by recent web browsers[2]. Emilé can save an XML or Extensible HTML document as standard HTML that can be served to older browsers.

However, XML is typically not served directly to the web browsers, but used as authoring format for more efficient web publishing. Advanced XML markup processors such as Interaction[3] and XPublish[4] generates standard HTML pages based on XML documents and style sheets. This gives you the flexibility of extensible markup while keeping your sites accessible for all web browsers.

This lesson introduces how to create new pages and author documents with Extensible HTML.

1. **Choose New from the File menu of Emilé.**
2. **From the Markup menu, select Document Types -> Extensible HTML -> HTML 4.0**

   Emilé inserts a document type declaration at the top of the file. This tells Emilé that the document type to use is Extensible HTML 4.0. Emilé will tailor its dialogs accordingly, providing menus and dialogs for efficient authoring with the specified document type.
3. **Paste or type in the following text below the document type declaration:**

```
Memo
Extensible Markup

From: Media Design in*progress
To: Webmasters
XML makes webmasters more efficient and flexible.
```

4. **Select the text “Extensible Markup”, then choose H1 from the Element submenu of the Markup menu. Click the Insert button of the resulting dialog.**

   The text is made into an H1 element, surrounded by an <H1> start tag and a </H1> end tag.
5. **Select the next line of text. Mark it up as a paragraph by choosing P from the Element submenu. Repeat for the two other lines in the document.**

   The three lines are now marked up as paragraphs. Continue until you have the following markup:

---

1. See http://www.w3c.org/markup for more about the next generation of HTML, Extensible HTML.
2. E.g. Internet Explorer 5.0 and later.
3. See http://interaction.in-progress.com for more about the Interaction web server companion.
4. See http://www.in-progress.com for more about the XPublish website publishing system.

```
<HTML>
   <HEAD><TITLE>Memo</TITLE></HEAD>
   <BODY>
      <H1>Extensible Markup</H1>
      <P>From: Media Design in*Progress</P>
      <P>To: Webmasters</P>
      <P>XML makes webmasters more efficient and flexible.</P>
   </BODY>
</HTML>
```

This example is both correct HTML and well-formed XML. Note how each start-tag has a corresponding end-tag. XML **requires** that all start tags have a correspondingly nested end-tag to specify when the element ends. In the example above, the <P> start-tags **starts** each paragraph, and the</P> end-tags ends a paragraph. In contrast, HTML often allow you to skip the end tag and in some cases also the start tag.

6. **Save the document as "memo1" in XML format.**

ABOUT THE MARKUP EDITOR

Emilé is based on a Macintosh clone of the powerful and widely used Emacs text editor.The editor provides numerous shortcuts that are the same as its UNIX/Windows counterpart for productive text editing. At the same time, Emilé has Macintosh ease of use and is specialized for markup, with menu driven editing tools, pane splitting, contextual menus, drag & drop, and more.

Lesson 2: Adding Custom Elements

Extensible Markup Language (XML) allows you to use custom tags to better markup your content. This lesson introduces how to use custom elements in a document.

In this lesson you will type the markup by hand. **Lesson 4: Declaring Custom Elements** will teach you how to customize Emilé to provide buttons and dialogs for the custom elements used in this lesson.

1. **Open the "memo1" document from the File menu of Emilé.**
2. **Remove all text before <BODY>.**

   This is optional. Keeping the HTML and HEAD element conforms with Extensible HTML, while removing these elements makes for clean XML.
3. **Remove the </HTML> at the end.**
4. **Replace the <BODY> start-tag with <MEMO> and the </BODY> end-tag with </MEMO>.**
5. **Replace the <H1> start-tag with <SUBJECT> and the </H1> end-tag with </SUBJECT>.**
6. **Make the second paragraph into a FROM element by substituting the <P> with <FROM> and </P> with </FROM>.**
7. **Make the third paragraph into a TO element by substituting the <P> with <TO> and </P> with </TO>.**

   You should now have the following content:

```
<MEMO>
   <SUBJECT>Extensible Markup</SUBJECT>
   <FROM>Media Design in*progress</FROM>
   <TO>Webmasters</TO>
   <P>XML makes Webmasters; more efficient and flexible.</P>
</MEMO>
```

   The markup above is the same memo as in **Lesson 1: Authoring Extensible HTML**, but with custom XML tags. The essence of XML is the ability to use custom tags to better describe the structure and meaning of your documents. Custom element types facilitates advanced processing and more efficient and flexible web publishing.
8. **Save the document.**
9. **Generate an HTML page from the document by choosing Save As from the File menu, selecting HTML for Format in the dialog and adding a ".html" suffix to the file name.**

   Emilé automatically adds HTML and HEAD elements if necessary, and converts the custom elements into standard HTML elements. The generated HTML elements will have a CLASS attribute, so that the look & feel of the document can be controlled with a CSS style sheet.

ABOUT XML AND EXTENSIBLE HTML

XML differs from HTML by describing the content and structure rather than the presentation. In descriptive markup, tags are used to declare the role of the various parts of the document, leaving the presentation to be designed in style sheets.

Extensible HTML is HTML that follows the syntax of XML, allowing the document to be extended with XML constructs and custom elements. If you already know HTML, there are only a few things you need to know in order to markup any content with Extensible HTML and XML:

- XML Markup **describes the document structure**, leaving the design of the presentation to style sheets.
- You can use **custom elements** to optimally describe the structure of your documents.
- An element always starts with a **mandatory start-tag**, e.g. `<PARAGRAPH>`, followed by content, ending with a corresponding **mandatory end-tag**, e.g. `</PARAGRAPH>`.
- **Elements can be nested** by enclosing other elements within an element.
- Enclosed elements must be **properly nested** (i.e. start-tags should close with matching end-tags).
- Custom elements should be **labeled using a noun** that describes the role of content.
- XML is **case sensitive** (i.e. it makes a difference whether an element identifier or attribute name is upper case or lower case). Use upper case in element identifiers and attribute names for backward compatibility with HTML.
- The start-tag of an element can contain attributes with characteristic qualities of the element. You can use **custom attributes**. Attributes should be nouns or adjectives that describe significant characteristics.
- An attribute has a name and a value with an equal sign in between. **Attribute values are always quoted.** Here is an example of an element with an attribute for language:
  `<ABSTRACT lang=”en”>This is an abstract in English</ABSTRACT>`
- Elements that doesn’t have any content are marked up as an **Empty Element**, with the same syntax as a start-tag but ending with ‘`/>`’ as in `<IMG SRC=”image.gif”/>`

This is almost all you need to know in order to author with extensible markup. XML has many features to formally declare your own markup constructs, but you don’t have to use these when authoring documents with XML.

Lesson 3: Declaring Internal Entities

An Entity is a unit of information and provides replacement text for an Entity Reference. An Entity Declaration advises about the existence of an entity, and either provides the replacement text or tells where the content can be found.

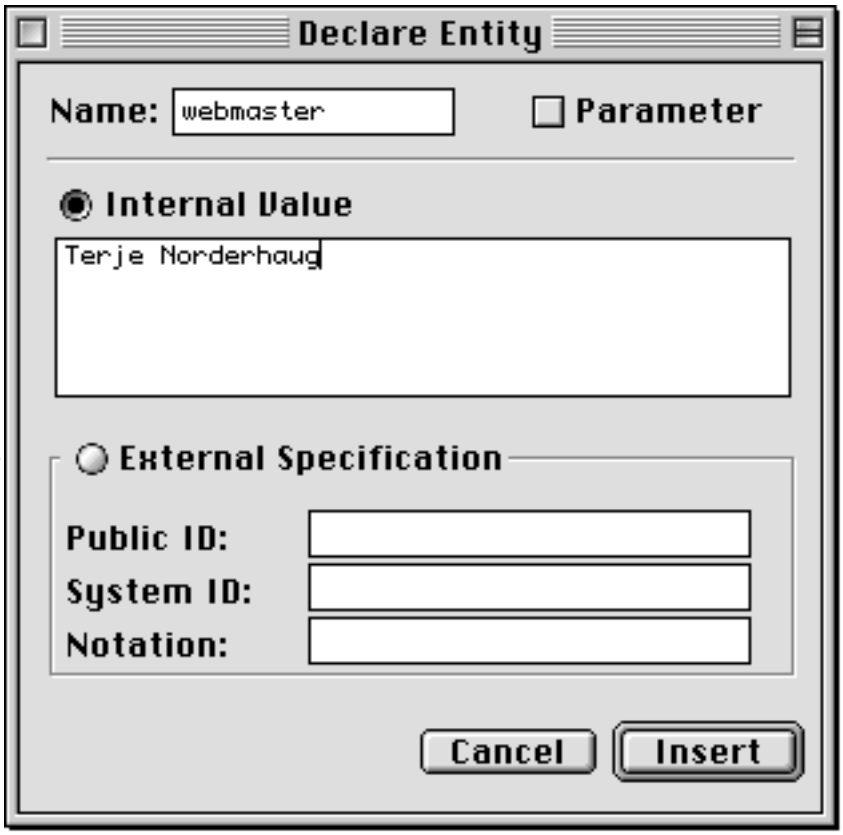
This lesson introduces how to declare Internal Entities that can be used in a specific document. Declaring such entities has several purposes, including:

- Customize the Entity Catalog palette with frequently used entities.
- Save typing by declaring internal entities for repeated content.
- Maintain frequently modified phrases as entities in the top of the document for easier updating of the content without the potential complications of search / replace.
- Specialize entities of the specified document type.

Here is how to define an internal entity called webmaster:

1. **Open the memo1 document from previous lessons using Open from the File menu.**
2. **Open the Declare Entity dialog from the Markup menu.**

   The Declare Entity dialog is used to declare a new entity or change an existing entity declaration. Selected text in a document will automatically become the default text for the entity.



3. **Type in lowercase "webmaster" in the field for Name**

   The entity name is used to reference the entity.

4. **Change the Internal Value to your name.**

   The internal value is replacement text for the entity. The replacement text will substitute any reference to the entity in the document.

5. **Click the Insert button to add the entity declaration in the document**

   Emilé will automatically place the entity declaration in a document type declaration in the beginning of the document. If the document doesn't have a document type

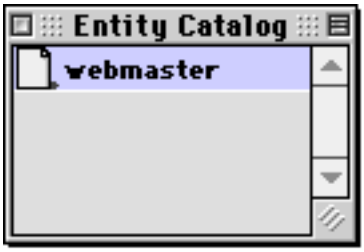
declaration, one will be created automatically before any other declarations are added.

The entity declaration should now appear similar to this:

```
<!DOCTYPE MEMO [
  <!ENTITY webmaster “Terje Norderhaug”>
]>
```

Note that the document type declaration may vary depending on the specified document type.

6. **Open the Entity Catalog from the Palettes of the Windows menu**



The Entity Catalog lists Entities that are specific for the document.

7. **Insert the entity reference &webmaster; in place of the two instances of the word “webmasters” in the document.**

Select the word “webmasters” in the document, and click on the item labeled Webmaster in the Entity Catalog[1].

8. **Save the Document**

Saving the document makes Emilé aware of the changes.

9. **Generate an HTML page by using Save As from the File menu.**

Type in a filename with a “.html” suffix in the Save As dialog, and select the radio button labeled HTML.

10. **Open the location in a web browser.**

All occurrences of the entity reference &webmaster; will have been replaced with the entity value.

1. You can alternatively select the entity in the Entities submenu of the Markup menu.

Lesson 4: Declaring Custom Elements

Recall from **Lesson 2: Adding Custom Elements** that elements are used to describe the role of various parts of the document. This lesson demonstrates how to declare custom element types. Formally declaring element types has several advantages, including:

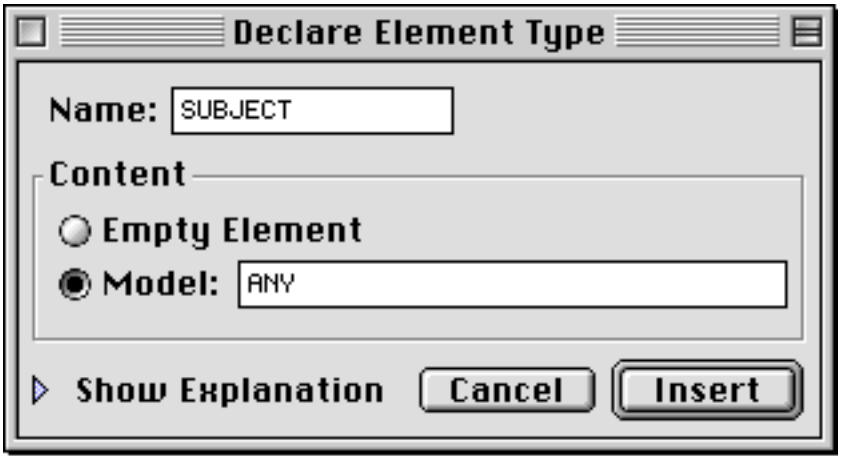
- Emilé can provide buttons and menus to insert custom tags and elements in the documents, saving the author from repeated typing.
- Emilé can provide dialogs to define the various attributes of an element, making authoring more efficient reducing the likelihood for mistakes.
- Emilé can provide on-line help advising the author about which attributes can be used in an element, as well as the structure of the content of the element.
- The document can be validated, warning about inconsistencies such as paragraphs within an header and other constructs that don't make sense.

In **Lesson 2: Adding Custom Elements** we introduced several custom elements to mark up the various parts of a memo. Now is time to formally define the elements:

1. **Open the memo1 document from the File menu.**
2. **Open the Declare Element dialog from the Markup menu**

   The Declare Element dialog facilitates the declaration of custom elements.

   Each element has a name and a content model that specifies the valid structure of the content of the element.



3. **Type the noun "SUBJECT" as the name for the element**

   The name is the same as in the tag `<SUBJECT>`.

4. **Check the radio button for Model, and type in ANY as content model.**

   ANY declares that the element can contain any other elements or content. XML allows you to specify which elements can be contained in an element so that Emilé can validate that the markup is correct.

5. **Click the Insert button to complete the dialog.**

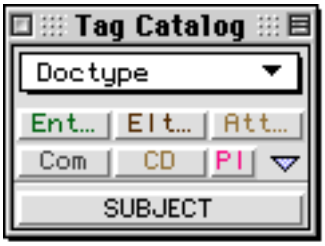
   The Element Declaration is inserted in the Document Type Declaration of the document, and should appear like this:

```
<!ELEMENT SUBJECT ANY>
```

To modify the element declaration, place the cursor in the declaration and select Inspect from the Markup menu. This will open the declaration in the same dialog that you originally used to declare the element.

6. **Save the document.**

   Saving the document updates Emilé about the new element type.



7. **Open the Tag Catalog from the Palettes of the Windows menu.**

   The new element type is listed in the Tag Catalog. Click the triangle in the bottom right of the catalog to view the button that inserts the element in the document.

8. **Repeat for the other elements of the memo.**

   The other elements are MEMO, TO, FROM and P.

ABOUT CUSTOM ELEMENTS

XML Elements should have names that describe content rather than presentation. This allows the presentation to be maintained in a separate style sheet, opens for advanced processing, and let the content be reused and processed by a wide range of software.

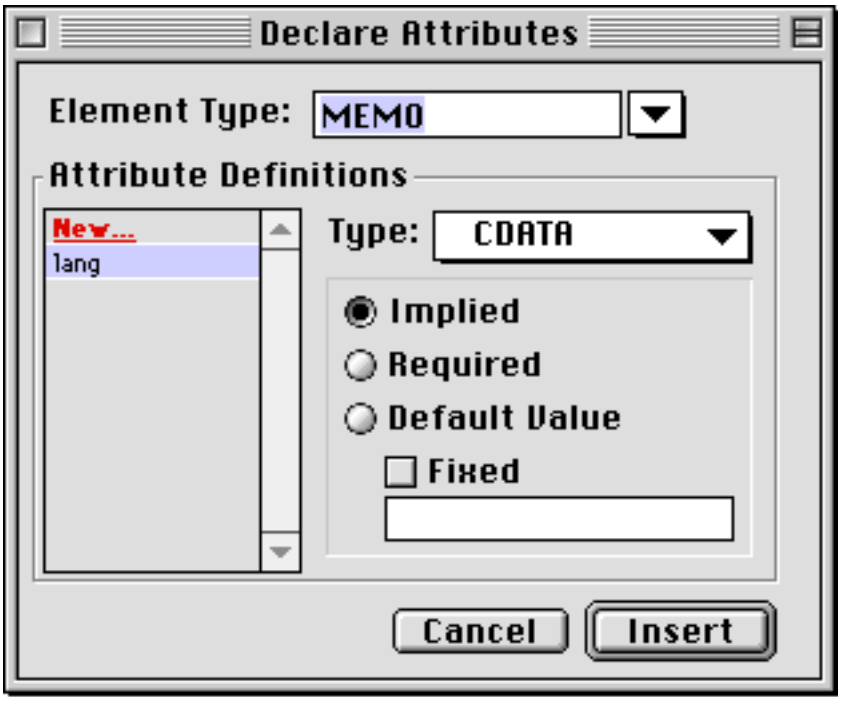
As a rule of thumb, **use nouns as the name of your custom elements**. Good names includes HINT, PERSON, and HEADER. Avoid names that represent colors, sizes, fonts or other presentational aspects. Also, avoid using verbs as element names, as mixing instructions or commands with markup makes your content less flexible for reuse.

Lesson 5: Declaring Attributes

Attributes are characteristic qualities of an element. The following steps will declare an attribute “lang” that can be used to specify the language of a memo:

1. **Select “Declare Attributes” from the Markup menu**

   A dialog opens with fields to specify the attributes of an element.



2. **Select MEMO as the Element Type for the attributes**

3. **Click New on the list of Attribute definitions, and give the new attribute the name “lang”.**

   The attribute definition list has an item for each attribute that can be used in an element.See XML documentation for further details about how to specify the attribute type and value.

4. **Click Insert to complete the attribute declaration**

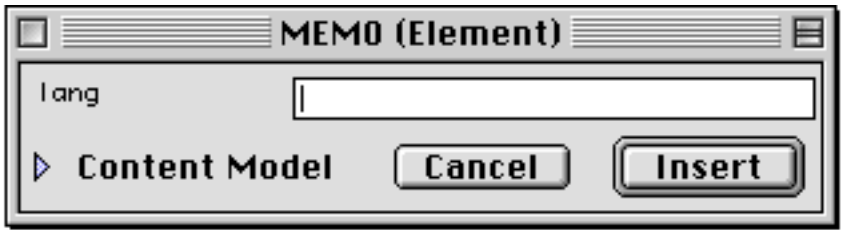
   An Attribute List Declaration is now inserted in the Document Type Declaration of the document. The document type declaration subset should now be as follows:

```
<!DOCTYPE MEMO [
   <!ENTITY webmaster "Terje Norderhaug">
   <!ELEMENT MEMO ANY>
   <!ELEMENT SUBJECT ANY>
   <!ELEMENT TO ANY>
   <!ELEMENT FROM ANY>
   <!ELEMENT P ANY>
   <!ATTLIST MEMO
             lang CDATA #IMPLIED>
]>
```

   The Attribute List Declaration can be modified by placing the cursor in the markup and select Inspect from the Markup menu. You can also change the declaration directly as long as the document is saved afterwards.

5. **Place the cursor in the <MEMO> start-tag of the MEMO element, and select Inspect from the Markup menu**

   This will bring up a dialog to configure the attributes of the element. There will be a field for language.



   Disclose the Content Model for information about valid content for the element. The explanation is created based on your custom content model.

6. **Type in a language in the field for "lang".**
7. **Click Insert to add the changes to the start tag of the element.**

Lesson 6: Defining a Document Type tells how you can use the same method to declare elements for multiple documents that uses the same markup.

Lesson 6: Defining a Document Type

Documents often have a similar structure. XML allows you to define document types that contain the elements and rules for describing the structure of a document. HTML is an example of a general document type with many elements and entities. Using XML, you can define document types that are specialized for your content.

The lesson is based on **Lesson 3: Declaring Internal Entities**, **Lesson 4: Declaring Custom Elements** and **Lesson 5: Declaring Attributes**. In those lessons you practiced how to declare document specific entities and elements. The same declarations can be used to define a document type, so that you don't have to repeat the same declarations in multiple documents.

In this lesson, you will create a simple Document Type Definition (DTD):

1. **Choose "New" from the Document Types submenu of the Markup menu.**

   Emilé displays a dialog to type in the name of a new document type.

2. **Type in "memo" as the name of your custom document type, and click the OK button of the dialog.**

   Emilé will create a new DTD file in its Document Types folder, and open it in the editor.

3. **Place the cursor at the end of the DTD file.**

4. **Select Declare Element from the Markup menu.**

5. **Type MEMO as the name of the element, type in `(SUBJECT, FROM, TO, P*)` for content model, and click the Insert button**

   An Element Declaration for HTML is now inserted in the cursor position of the Document Type Definition.

   The content model declares what is valid content for elements of the type. The content model above declares that a MEMO should contain a sequence of the elements SUBJECT, TO, and FROM, followed by any number of paragraphs.

   The star '*' after the P means that a paragraph element can occur zero or more times. A plus '+' in the same position would signify that a paragraph can occur once or more. A question-mark '?' would make the element optional.

   The different items in the content model for MEMO are separated by commas, which signifies that they should occur sequentially. They can alternatively be separated by a bar (' | ') which signifies that the items can occur in any order.

6. **Select Declare Element from the Markup menu**

7. **Type SUBJECT as the name of the element, type in (#PCDATA) as content model, and click the Insert button**

   The #PCDATA represents character data. Character data is regular text without any markup. Thus, the content model for SUBJECT states that it can only contain plain text and no markup.

8. **Repeat to declare the FROM and TO elements to have (#PCDATA) as content model**

9. **Declare an element P with ANY as content model**

   The content model ANY means that the element can contain any element or data.

   The DTD should now be as follows:

```
<?xml version="1.0"?>
<!ELEMENT MEMO (SUBJECT, FROM, TO, P*)>
<!ELEMENT SUBJECT (#PCDATA)>
<!ELEMENT FROM (#PCDATA)>
<!ELEMENT TO (#PCDATA)>
<!ELEMENT P ANY>
```

   To modify any of the declarations, place the cursor in the markup and select Inspect from the Markup dialog.

10. **Close the editor, saving the Document Type Definition**

    Saving the DTD will make Emilé aware of the changes. The name of the document type should now be listed in the Document Types submenu of the markup menu.

USING THE MEMO DOCUMENT TYPE

You are now ready to make use of the memo document type.

11. **Create a new document using New from the File menu of Emilé.**

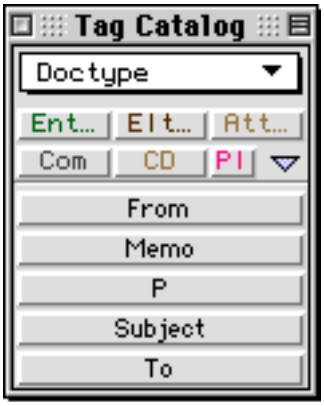
12. **Declare the document type by choosing "memo" from the Document Types sub-menu of the markup menu.**

    The prolog of the document now looks as follows:

```
<?xml version="1.0"?>
<!DOCTYPE name SYSTEM "memo">
```

13. **Open the Tag Catalog from the Palettes of the Windows menu (if not already shown)**

    The tag catalog can also be opened using the <> button in the upper right corner of the editor.



14. **Click the small triangle in the Tag Catalog to disclose the element buttons for the document**

    The Tag Catalog lists the same elements as you included in the document type definition.

15. **Place the cursor on the first line after the Document Type Declaration. Click the MEMO button of the Tag Catalog to insert the root element. Use the other buttons to build a memo like the one in Lesson 2: Adding Custom Elements.**