Introduction to XML

What markup languages have you used (or looked at) (or heard of)?

What markup languages have you used (or looked at) (or heard of)?

- (X)HTML (Web pages)
- EAD (archival finding aids)
- DocBook (books, such as manuals)
- TEI (texts, scholarly editions)
- MEI (music)

What’s the function of a mark up language?

Why do specialized markup languages exist in the sciences, humanities, math, music, etc.?

Why do scholars and professionals in many different disciplines find it useful to mark up, or encode, information?

What’s the function of a mark up language?

To make explicit certain features of text in order to aid the processing of that text by computer programs.

We encode texts because plain text isn’t good enough (*for what we want to do*)

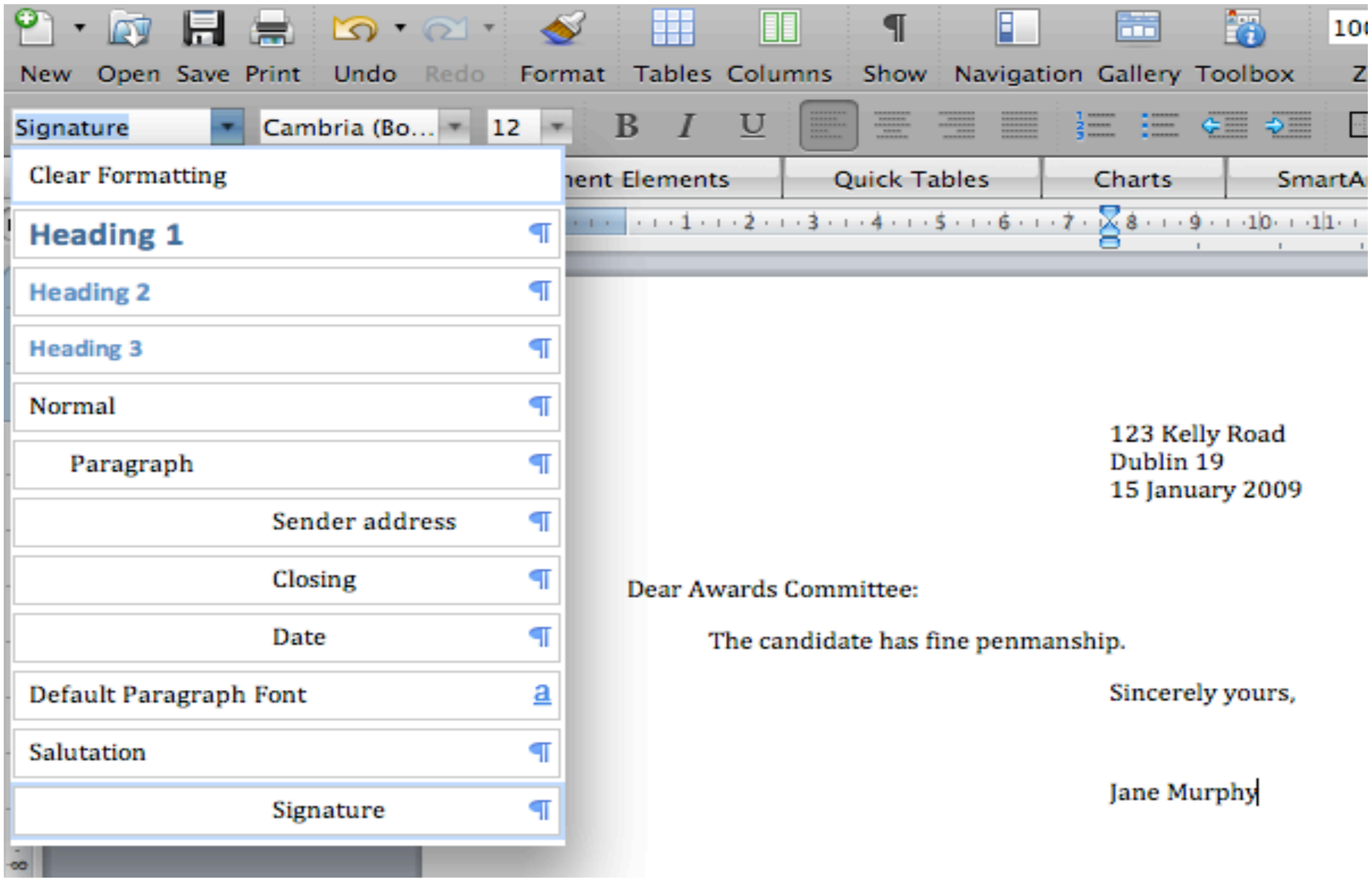
What if you want to...

```
123 Kelly Road
Dublin 19
15 January 2009

Dear Awards Committee:

The candidate has fine penmanship.

Sincerely yours,

Jane Murphy
```

Publish a collection of letters and decide after beginning that you want to have the sender’s address and closing always right-aligned?

Search your collection of letters to extract a list of all senders and another list of all recipients?

Word processor styles: Encoding under the surface



123 Kelly Road
Dublin 19
15 January 2009

Dear Awards Committee:

The candidate has fine penmanship.

Sincerely yours,

Jane Murphy

Extensible Markup Language (XML): word processor styles on steroids

Can have one style inside another (‘nesting’)

- There's a title in this citation!
- There's a quote in this paragraph!

Can give properties to these styles, e.g.,

- This salutation is formal.
- This sentence is sarcastic.
- This word is misspelled.

XML in brief (1)

Open, non-proprietary standard

Stored in plain text but usually thought of as contrasting with it (as above)

Marks beginning and ends of spans of text using tags:
<sentence>This is a sentence.</sentence>

XML in brief (2)

Spans of text must nest properly:

Wrong:

```
<sentence>Overlap is <emphasis>not allowed!</sentence></emphasis>
```

Right:

```
<sentence>Overlap is <emphasis>not allowed!</emphasis></sentence>
```

Elements (tags), attributes, values, content

<sentence type=“declarative”>This is a sentence.</sentence>

<sentence type=“interrogative”>Is this is a sentence?</sentence>

Elements (tags), attributes, values, content

Elements may have one attribute, many attributes, or none, but each attribute on any given element must be unique.

Valid: one attribute

```
<sentence type=“declarative”>This is a sentence.</sentence>
```

Valid: two unique attributes

```
<sentence type=“interrogative” xml:lang=”en”>Is this is a sentence?</sentence>
```

Valid: no attributes at all

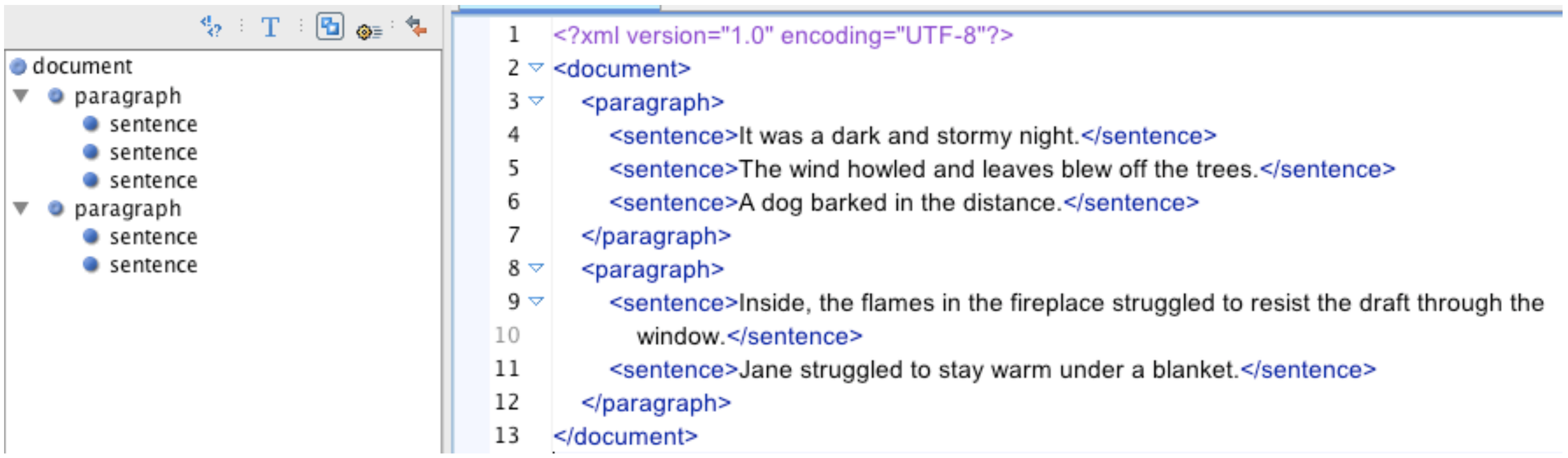
```
<sentence>This is a sentence.</sentence>
```

Invalid: two instances of the same attribute (type=) on one element

```
<sentence type=“declarative” type=“true”>This is a sentence.</sentence>
```

XML as a tree

- We use family tree terms: parent, child, sibling, ancestor, and descendent.
- *Remember, everything must nest properly!*



```
<?xml version="1.0" encoding="UTF-8"?>
<document>
  <paragraph>
    <sentence>It was a dark and stormy night.</sentence>
    <sentence>The wind howled and leaves blew off the trees.</sentence>
    <sentence>A dog barked in the distance.</sentence>
  </paragraph>
  <paragraph>
    <sentence>Inside, the flames in the fireplace struggled to resist the draft through the
      window.</sentence>
    <sentence>Jane struggled to stay warm under a blanket.</sentence>
  </paragraph>
</document>
```

Wait, this all looks a lot like HTML!

HTML is a specific implementation of XML (well, actually, its predecessor SGML) that has pre-determined elements and attributes.

You can't create your own elements, so its usefulness is limited.

Schemas (DTDs and others)

A syntax for your XML documents, specifying:

- Which elements are allowed
- Which elements may nest inside of others
- In what order these elements must occur
- How many times they may repeat
- What attributes they may have
- What values those attributes may have

http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ST.html#STIN

Why would you want to constrain your document structure like this?

- Prevent errors in creating the XML
- Make it easier to search the text

Remember we were going to extract names of senders and recipients? Conforming to a schema helps you know where to expect to find them within your XML documents.

Structure, not appearance

Most people use XML to describe the structure of a document rather than its appearance. Information about how to render various components of the document is usually stored separately, in a *stylesheet*.

But how do we...

- Know what element and attribute names to use?
- Make decisions about defining and constraining our document structure?
- Avoid reinventing the wheel, and build on work that's already been done?
- Ensure that our texts can be understood and used by others?

Use something that already exists!

TEI

TEXT ENCODING INITIATIVE

http://www.tei-c.org/index.xml

http://www.tei-c.org/Guidelines/P5/index.xml

Questions?