# JITTs

# (Just-In-Time-Trees)

Patrick Durusau, pdurusau@emory.edu

*Society of Biblical Literature*

Matthew Brook O'Donnell, matt@opentext.org

*Publishing Dimensions & OpenText.org*

jitts.org

# Hang'em High:
## Capturing Desperate Markup Trees in the Wild!

## *Question #1*

**Is Overlap Important for Markup?**

jitts.org

## Example: print vs. logical structure

- SGML (ISO 8879) uses this as example of use of CONCUR
- Now have XSLT but presentation is only one aspect of multiple structures
- What about overlapping or differing analysis of texts?
- Key is multiple structures, for any reason.

# Example: Versioning

- Versioning is a special case of overlap
- Needed for:
  - Contracts
  - Legislative documents
  - Publications
  - Treaties
- Can you say “Enron?”

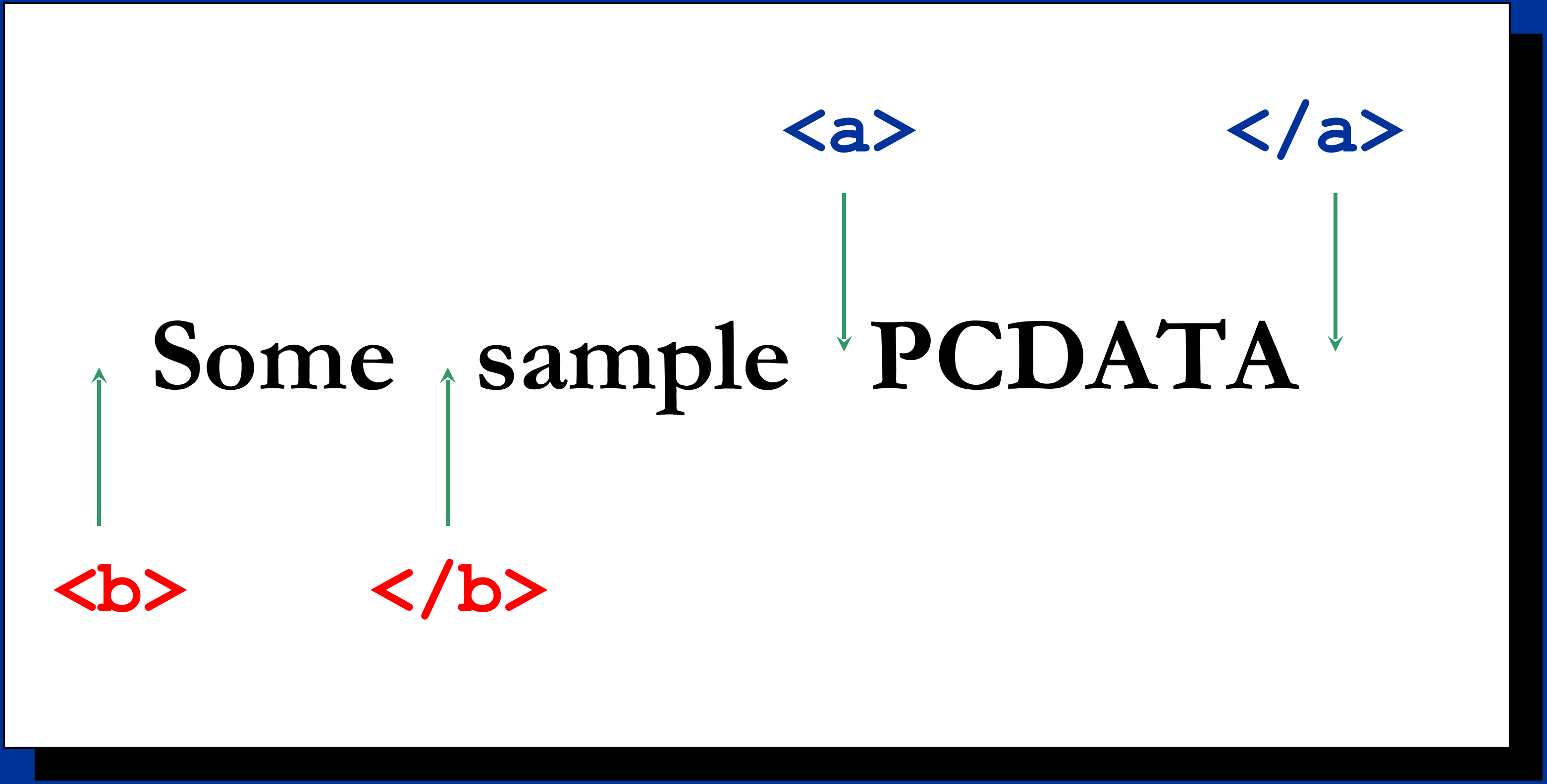
## *Observation #1*

**Overlap is important because documents are complex**

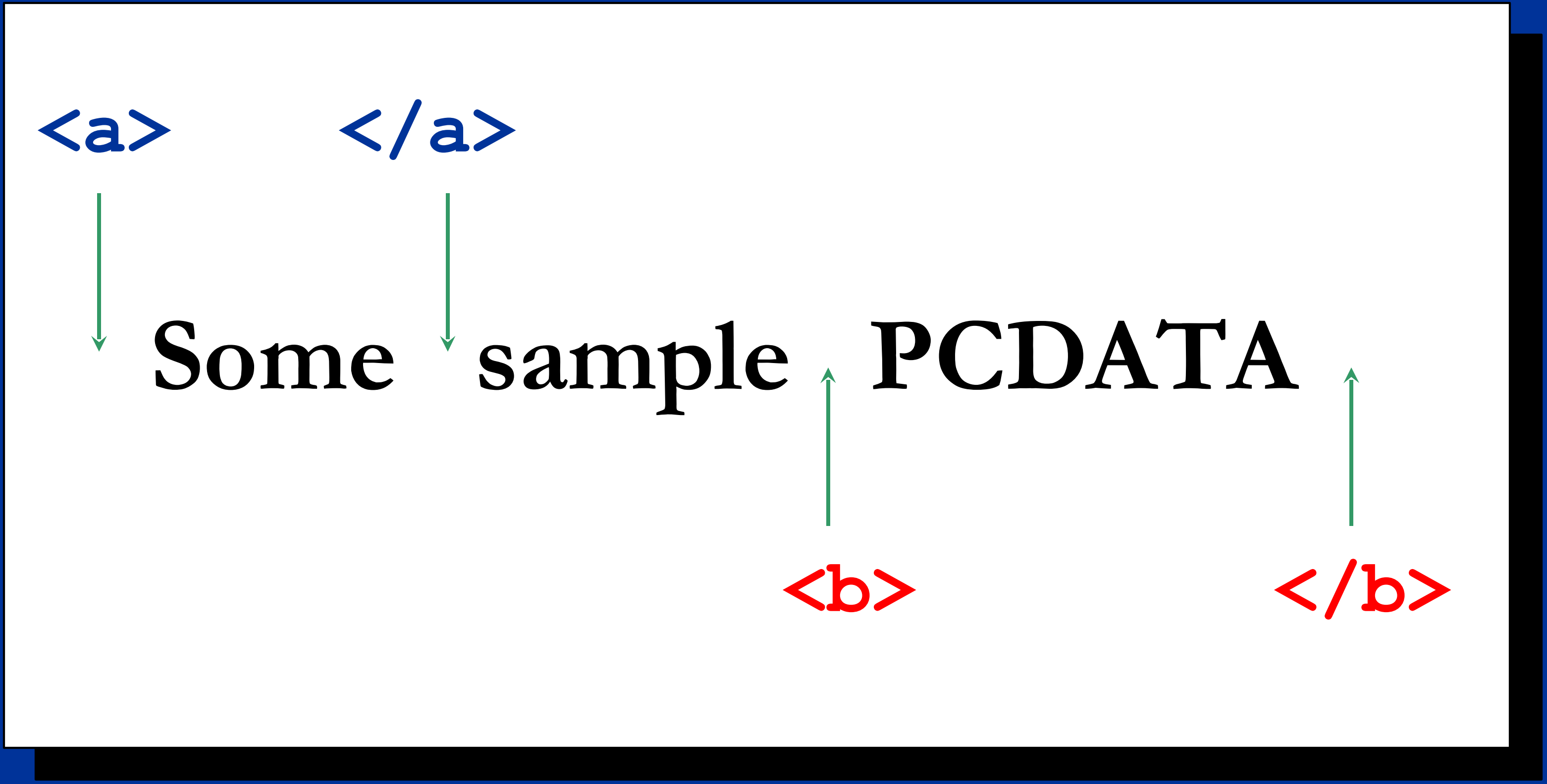
## Interlude #1

# Conversing with the trees!

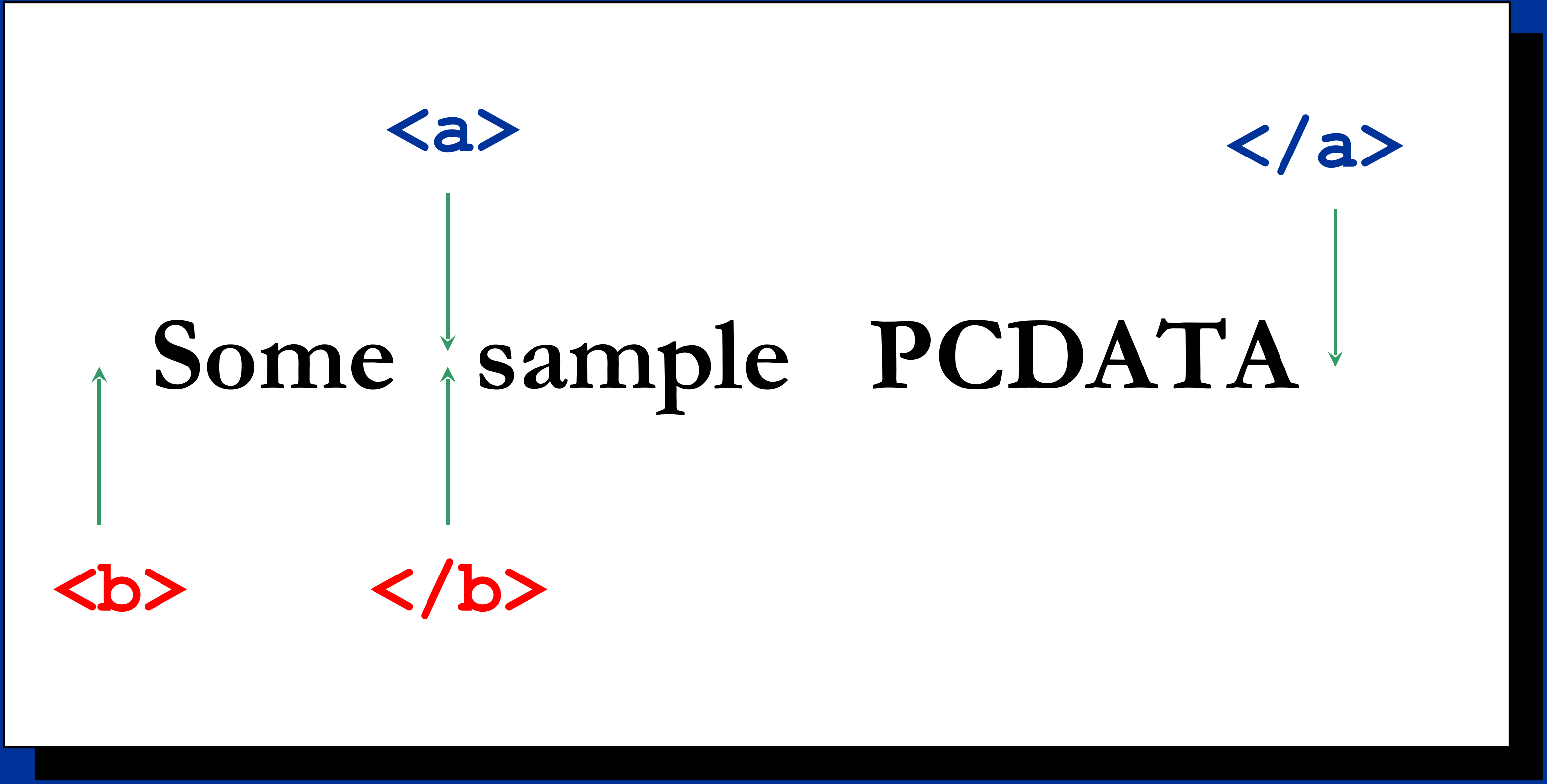
## Case 1: No overlap



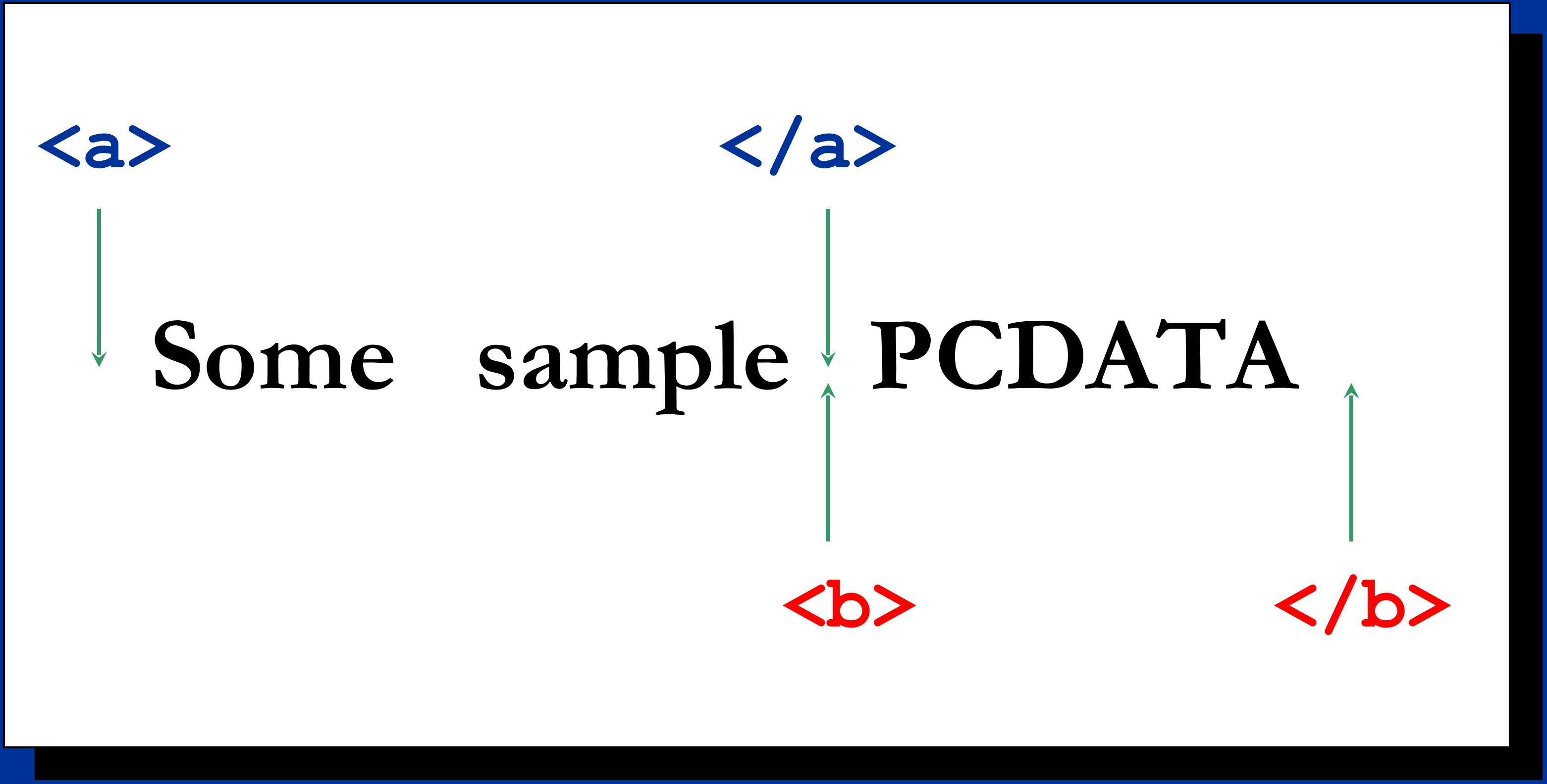
# Case 13: No overlap



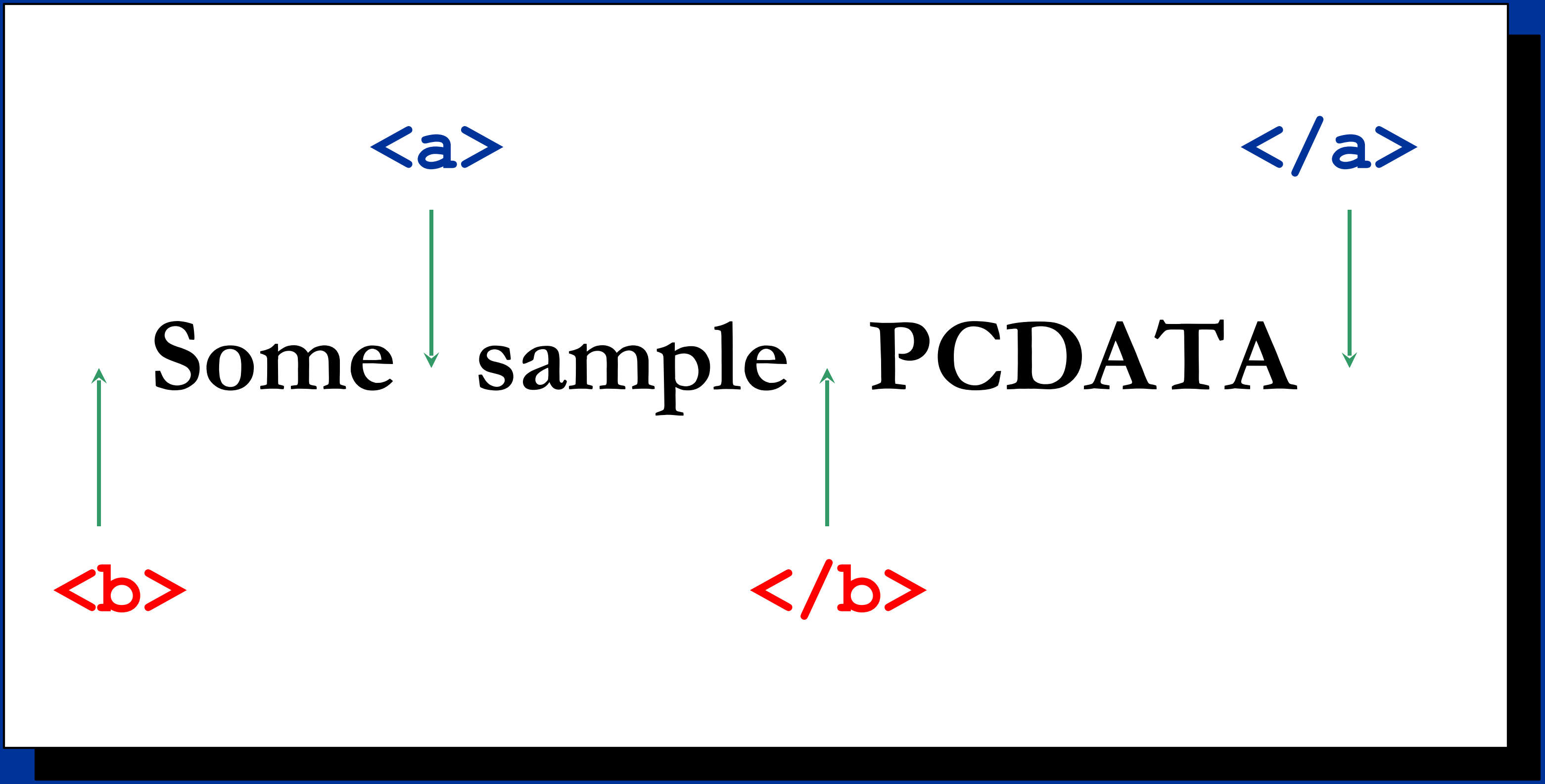
# Case 2: Shared start/end point



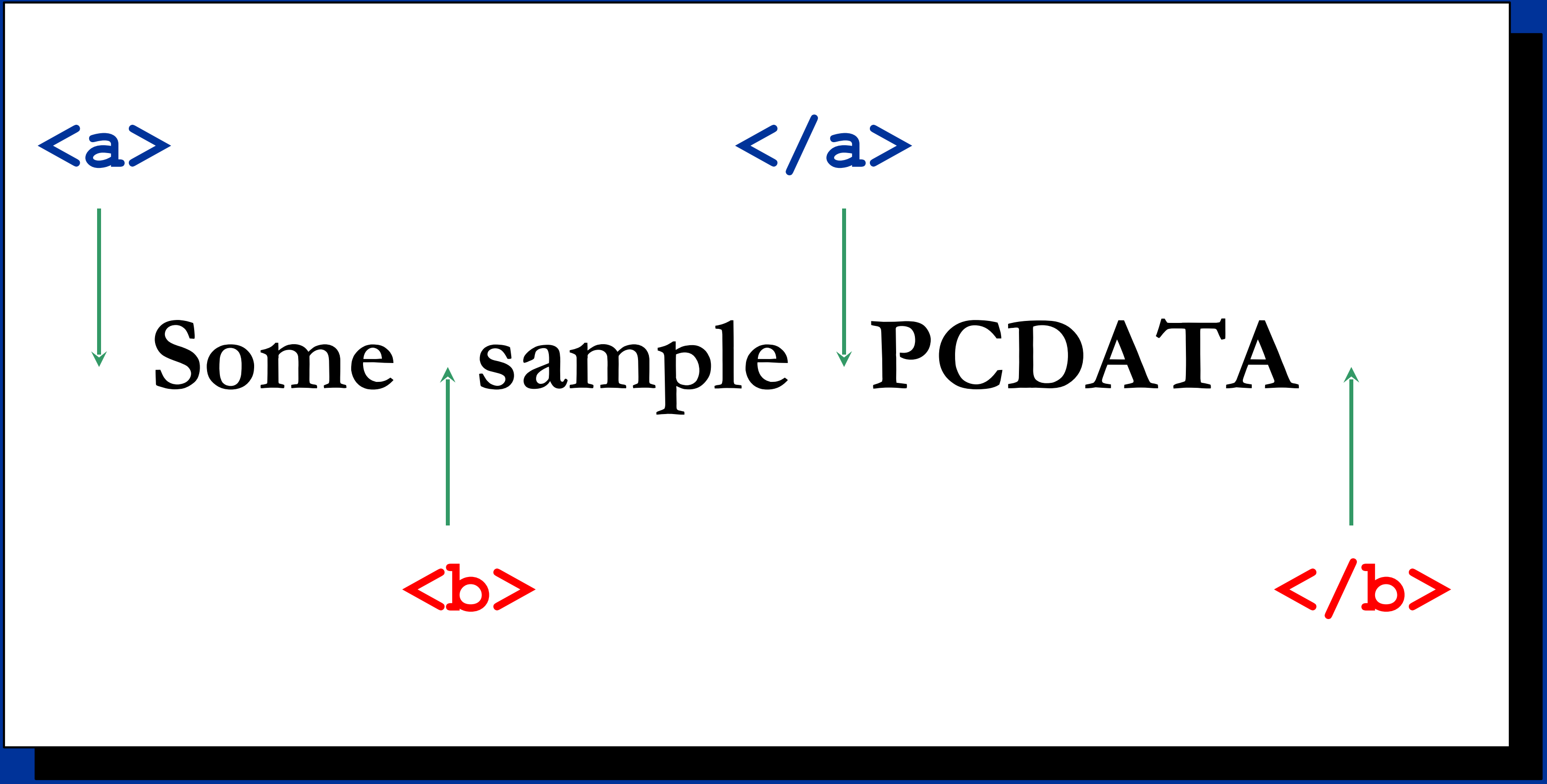
# Case 12: Shared start/end point



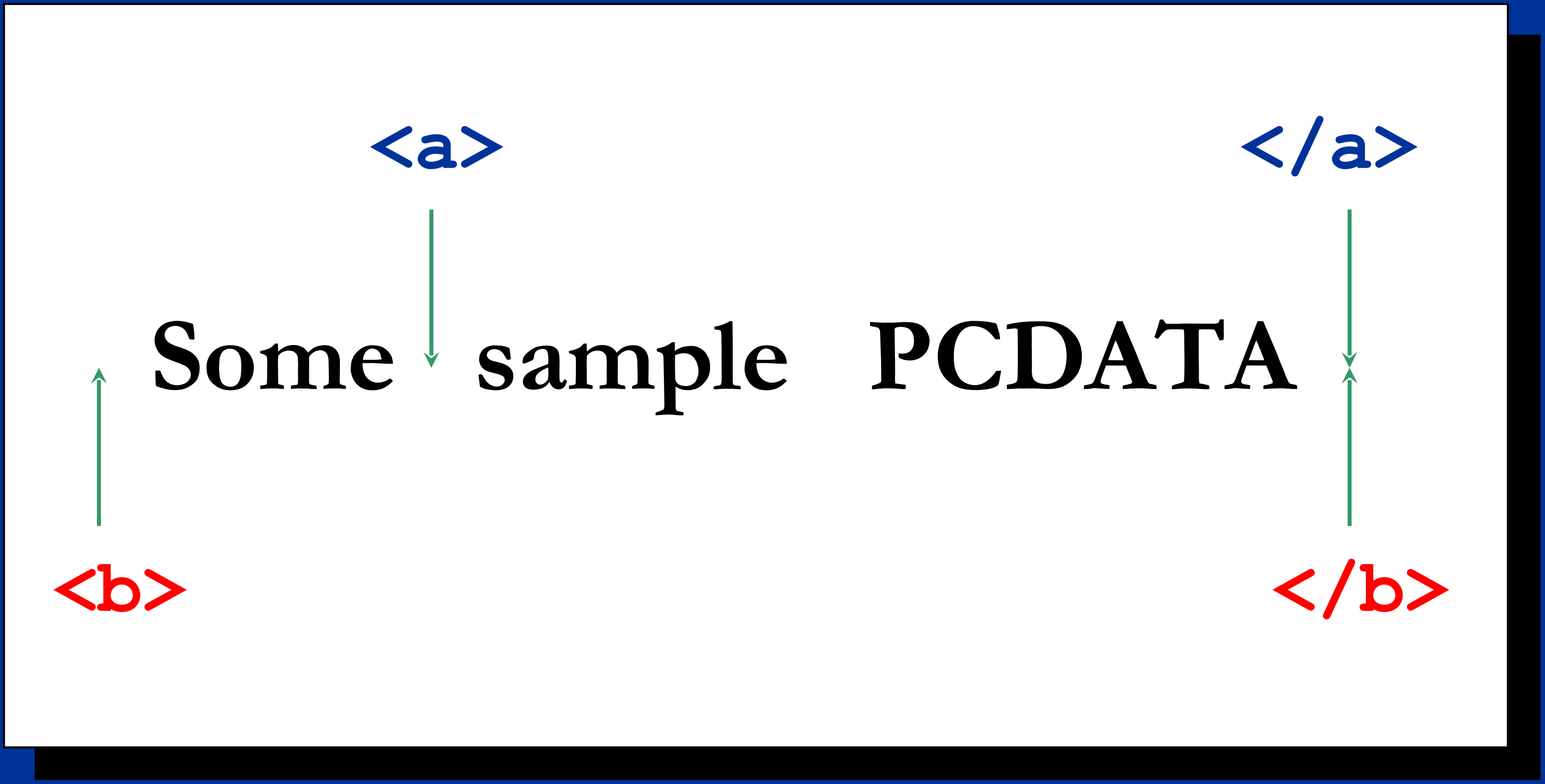
## Case 3: ‘Classic’ overlap



# Case 11: ‘Classic’ overlap

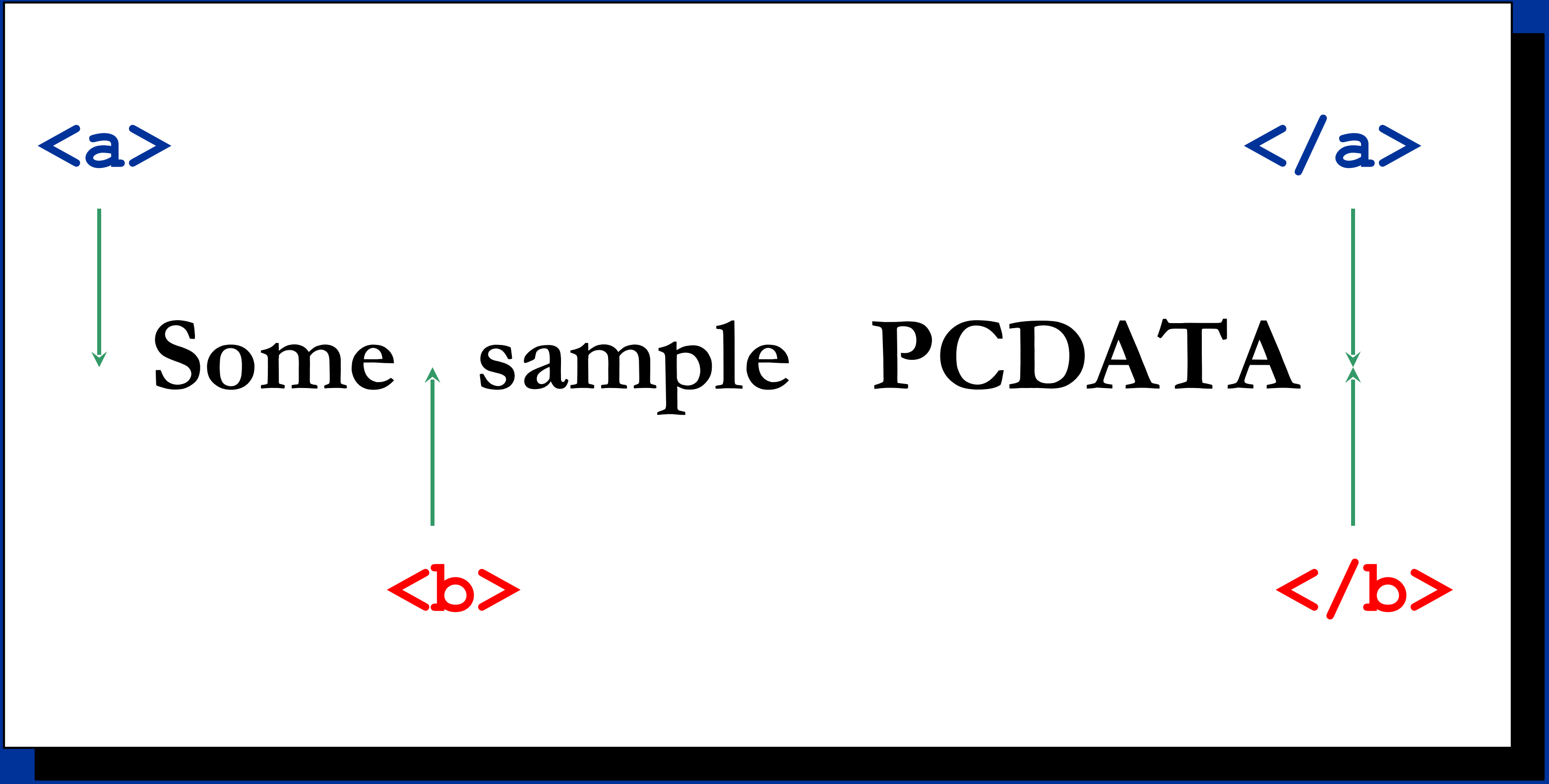


# Case 4: Shared end point

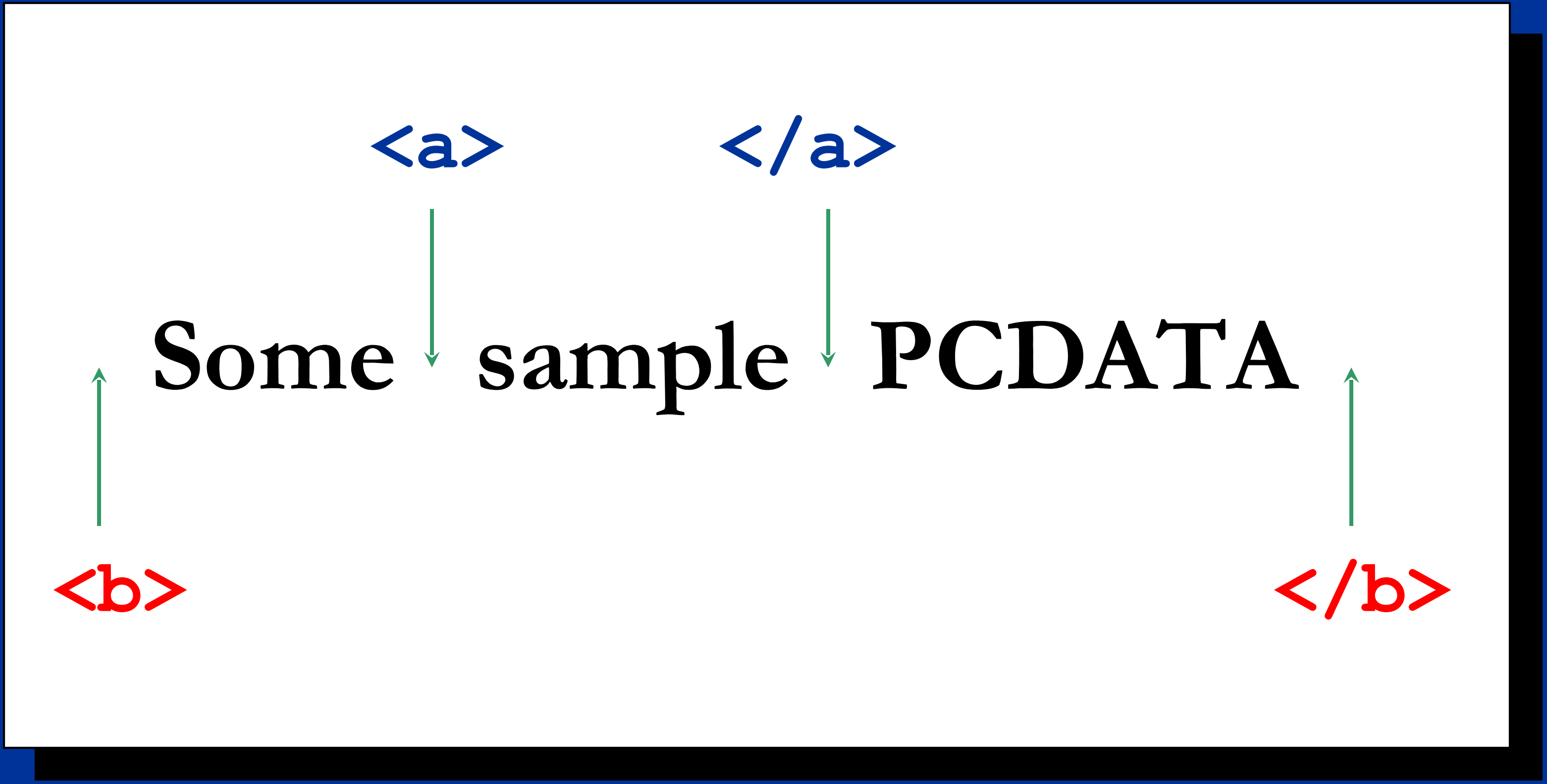


jitts.org

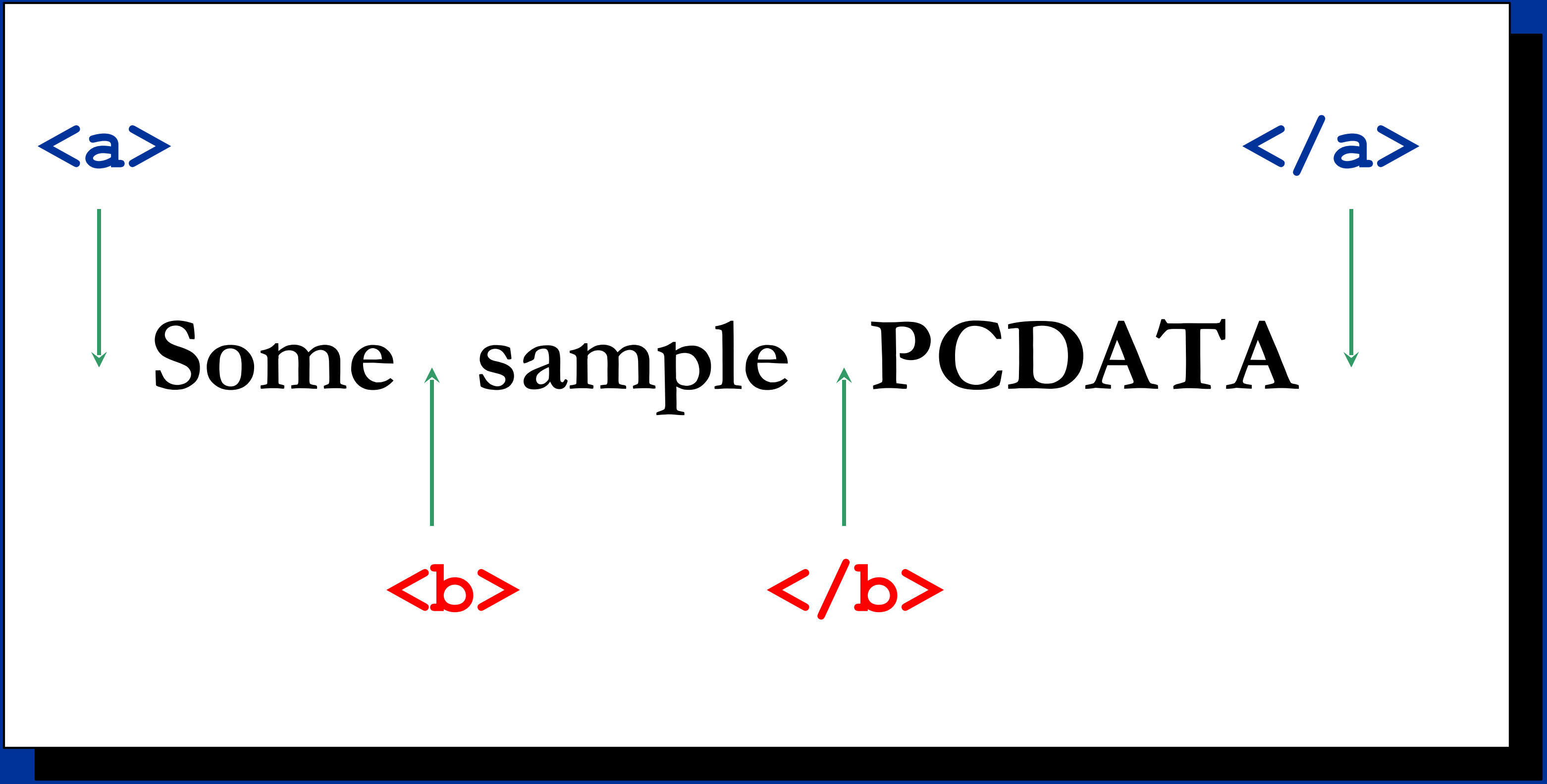
# Case 10: Shared end point



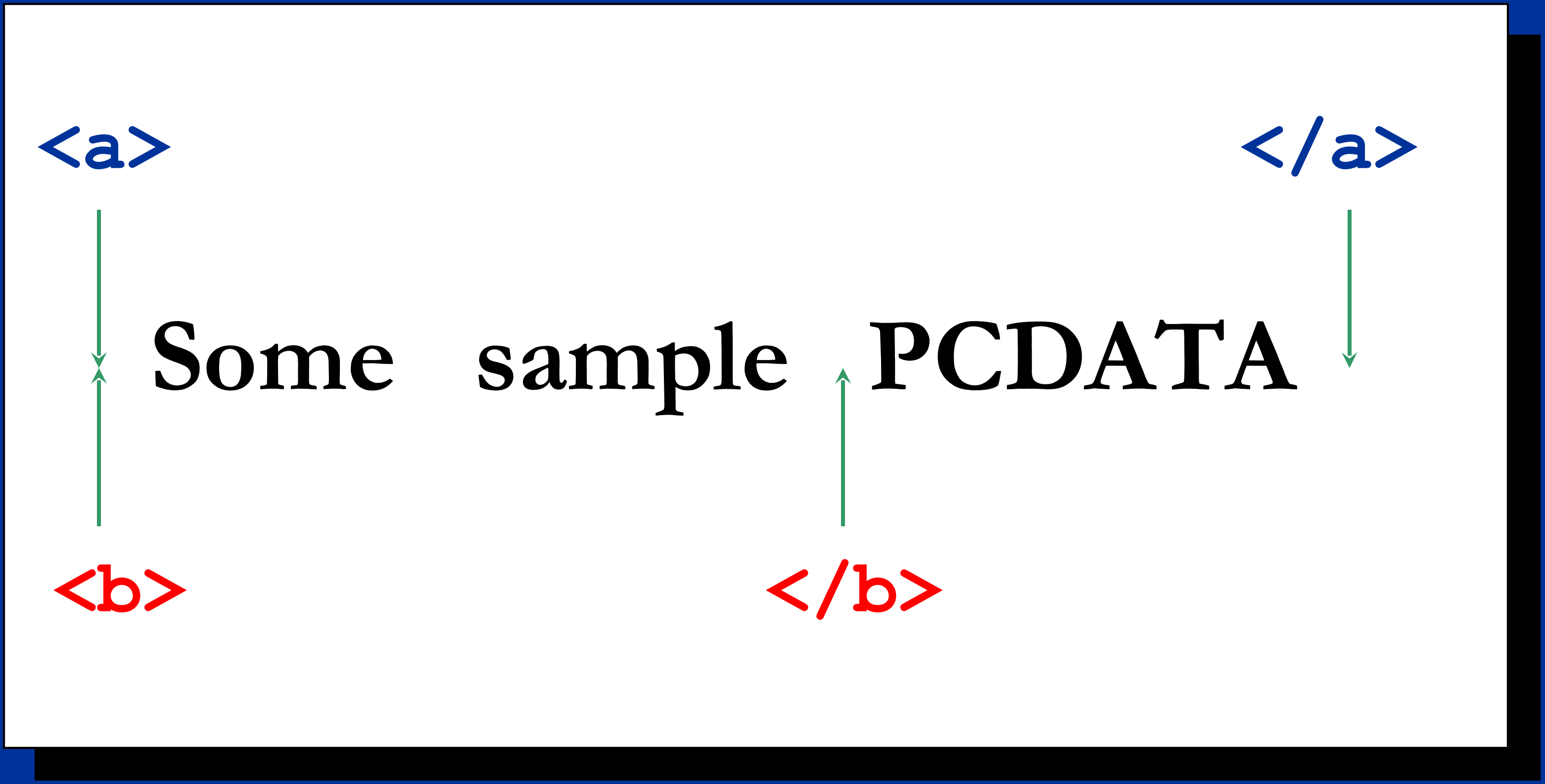
# Case 5: One element contains other



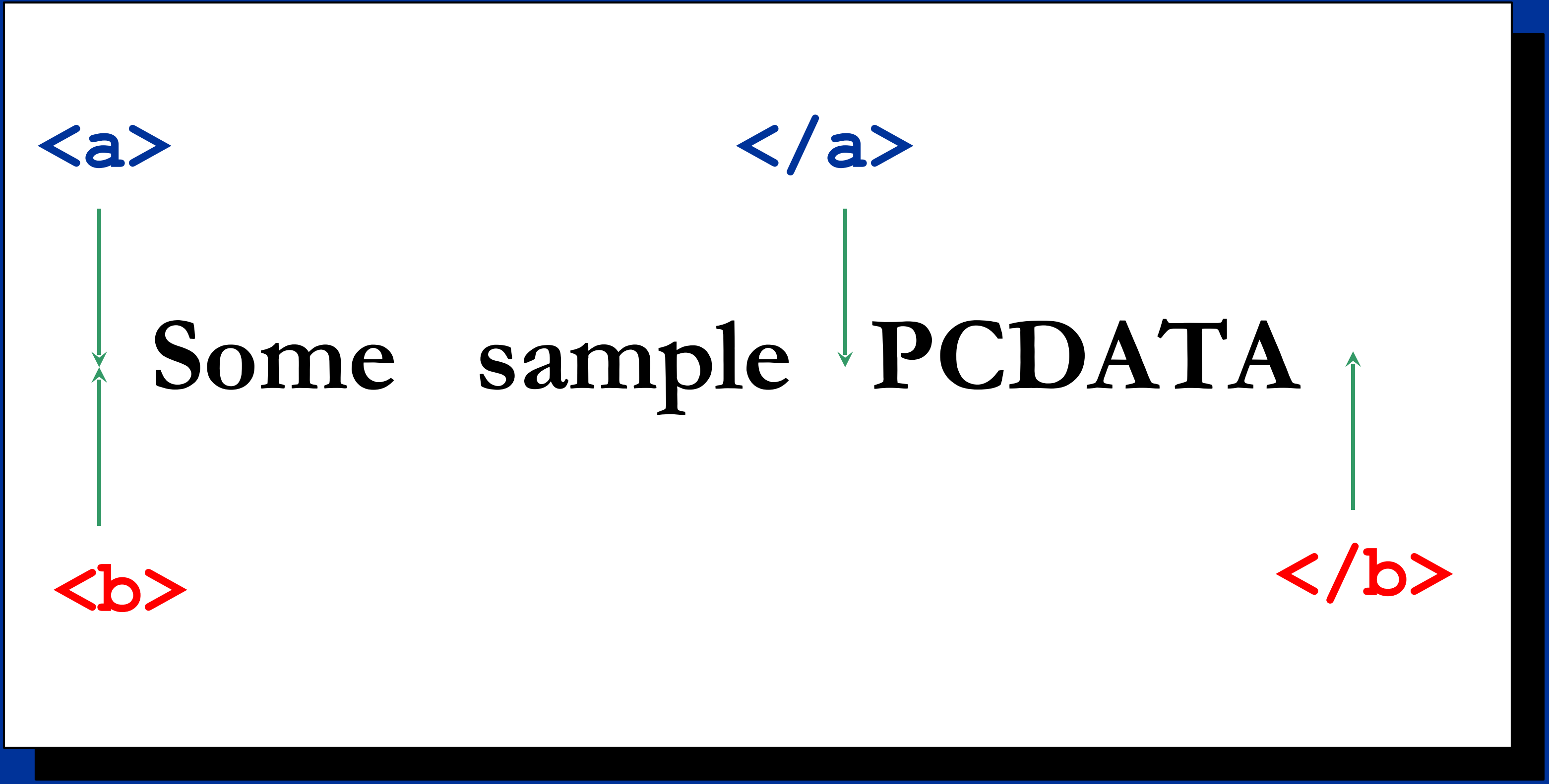
# Case 9: One element contains other



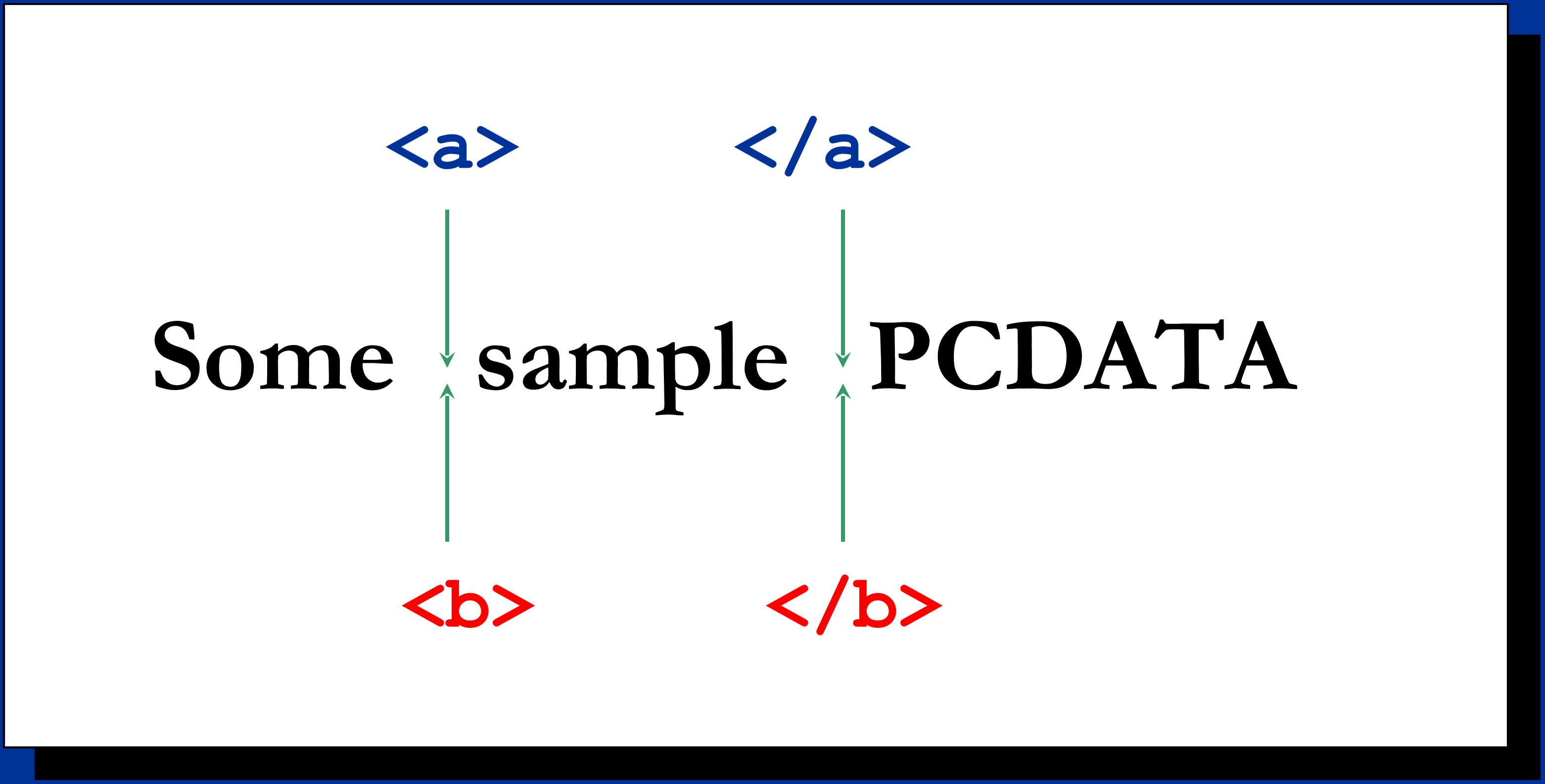
# Case 6: Shared start point



# Case 8: Shared start point



# Case 7: Shared start and end points



# Cases 1, 5, 9 & 13 handled by traditional markup

```
<b>Some</b>
sample
<a>PCDATA</a>
```

```
<b>Some
<a>sample</a>
PCDATA</b>
```

```
<a>Some</a>
sample
<b>PCDATA</b>
```

```
<a>Some
<b>sample</b>
PCDATA</a>
```

Tree model imposes parent-child relationship here

jitts.org

## *Observation #2*

**Trees and texts are both complex**

## Question #2

So where is the problem with markup?

## SGML Markup

- single tree
- conflates lexical and syntactical levels of language
- allows CONCUR (but as multiple static trees)
- HyTime (note 553) requires separate grove for each tree (heavy processing demand)

# XML Markup

- Discarded CONCUR
- Well-formedness – required with or without DTD/Schema
- Continues conflation of lexical and syntactical levels of markup
- Results in markup trees that don’t fit texts!

## But nature of text necessitates a solution..

- Milestones
- Segmentation
- Stand-off markup
- Non-XML syntaxes (Mecs, TexMecs, LMNL)

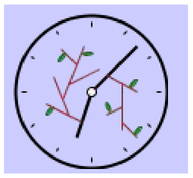
# Example: *Paradise Lost*

```
<line>
      <seg> Of Man's first
disobedience,</seg>
      <seg> and the fruit
</line>
<line> Of that forbidden tree whose
mortal taste
</line>
<line> Brought death into the World,
      </seg>
</line>
```

# Milestones

```
<div type="segment">

      <seg id="s1"/>

      <line n="Bk.1.1">Of Man's first
disobedience, <seg corres="s1"/>

      <seg id="s2"/> and the fruit</line>

      <line n="Bk.1.2">Of that forbidden tree
whose mortal taste</line>

      <line n="Bk.1.3">Brought death into the
World, and all our woe,</line>

      <seg corres="s2"/>

…
</div>
```

- Empty elements used to mark locations in the text
- Correspondence attributes provide minimal utility
- Require special processing
- Offers no validation



# Segmentation

```
<div type="segment">

      <line n="Bk.1.1"><seg id="s1">Of Man's
first disobedience, </seg>
      <seg id="s2a" next="s2b"> and the
fruit</seg></line>

      <line n="Bk.1.2">
      <seg id="s2b" prev="s2a" next="s2c"> Of
that forbidden tree whose mortal
taste</seg></line>
      <line n="Bk.1.3">
      <seg id="s2c" prev="s2b">Brought death
into the World, and all our   woe,</seg></line>…
</div>
```

jitts.org

# Stand-off markup

```
<div type="segment">
      <w id="w1">Of</w>
      <w id="w2">Man's</w> <w id="w3">first</w>
      <w id="w4">disobedience,</w>
      <w id="w5">and</w> <w id="w6">the</w>
<w id="w7">fruit</w>
      …
</div>
```

__Segments:__

```
<seg id="s1" start="#w1" end="#w4"/>
<seg id="s2" start="#w5" end="#w23"/>
```

__Lines:__

```
<line id="l1" start="#w1" end="#w7"/>
<line id="l2" start="#w8" end="#w14"/>
<line id="l3" start="#w15" end="#w23"/>
```

jitts.org

## Mecs, TeXMecs

- variant syntaxes for Wittgenstein Project
- no processing tools
- never adopted beyond this project
- but very useful for insights into the nature of overlap and the requirements of non-trivia texts

## LMNL

- based on core range algebra
- records structural information based on location in string (text as string)
- allows unlimited overlapping hierarchies
- requires specialized syntax and conversion of documents

# Limitations

- lack of processing tools
- variant syntaxes require conversion
- loss of easy access to tree API to text
- preserve lexical/syntactical confusion of SGML/XML (except Mecs,TeXMecs, LMNL)

## *Question #3*

**What is markup?**

## *Observation #3a*

**Markup is metadata about PCDATA**

## *Observation #3b*

*…though typically recorded inline with tree notation*

**Metadata can be recorded on each PCDATA item instead of around it**

## Bottom-Up Virtual Hierarchies (BUVH)

•divorces membership in a hierarchy from markup in a document

• membership in a hierarchy is recorded as an XPath expression

• membership information recorded for each atom of PCDATA

• allows validation of multiple hierarchies

• allows querying across multiple hierarchies

# Example: *Paradise Lost* line hierarchy

```
<div type="book">
  <div type="segment">
    <line>Of Man's first disobedience, and the fruit</line>
    <line>Of that forbidden tree whose mortal taste </line>
```

/div[1][@book='name']/div[1][@type='segment']/line[1]/*[1]

```
    …
  </div>
</div>
```

jitts.org

## Example: *Paradise Lost* clause hierarchy

```
<div type="book">
  <div type="segment">
    <seg>Of Man's first disobedience, </seg>
    <seg> and the fruit  Of that forbidden tree whose mortal
```

/div[1][@book='name']/div[1][@type='segment']/seg[1]/*[2]

```
    …
  </div>
</div>
```

jitts.org

# Example: *Paradise Lost*

```
<w ln:text=“/div[1][@book=‘name’]/div[1][@type=‘segment’]/line[1]/*[1]”
   sg:text=“/div[1][@book=‘name’]/div[1][@type=‘segment’]/seg[1]/*[1]”
>Of</w>

<w ln:text=“/div[1][@book=‘name’]/div[1][@type=‘segment’]/line[1]/*[2]”
   sg:text=“/div[1][@book=‘name’]/div[1][@type=‘segment’]/seg[1]/*[2]”
>Man's</w>

<w ln:text=“/div[1][@book=‘name’]/div[1][@type=‘segment’]/line[1]/*[3]”
   sg:text=“/div[1][@book=‘name’]/div[1][@type=‘segment’]/seg[1]/*[3]”
>first</w>

<w ln:text=“/div[1][@book=‘name’]/div[1][@type=‘segment’]/line[1]/*[4]”
   sg:text=“/div[1][@book=‘name’]/div[1][@type=‘segment’]/seg[1]/*[4]”
>disobedience,</w>
```

# Example: *Paradise Lost*

```
<w ln:text="/div[1][@book='name']/div[1][@type='segment']/line[1]/*[5]"
    sg:text="/div[1][@book='name']/div[1][@type='segment']/seg[1]/*[1]"
>and
```

Classic overlap

```
<                                 e']/div[1][@type='segment']/line[1]/*[6]"
                                  e']/div[1][@type='segment']/seg[2]/*[1]"
>

<                                 e']/div[1][@type='segment']/line[1]/*[7]"
                                  e']/div[1][@type='segment']/seg[2]/*[2]"
>
```

```
…
<seg>
        and the fruit
</line>
<line>
        Of
…
```

```
<w ln:text="/div[1][@book='name']/div[1][@type='segment']/line[2]/*[1]"
    sg:text="/div[1][@book='name']/div[1][@type='segment']/seg[2]/*[4]"
>Of</w>
```

## BUVH: Analysis

- illustrates localized nature of overlap
- requires additional processing of texts
- results in redundant information

## Question #4

What is the problem?

Isn't it is a question of recognizing the tree we want?

jitts.org

## JITTs: Just-In-Time-Trees

- Trees are imposed on texts
- Elements of a tree declared
- Content models declared
- Vocabulary of markup declared
- Non-elements, ignore/remove

## JITTs: Processing Model

- Declare tree (elements)
- Parse texts
- Declared elements = events
- Declared tree must be well-formed and valid
- Non-declared elements, pass or discard

## JITTs Filter Process

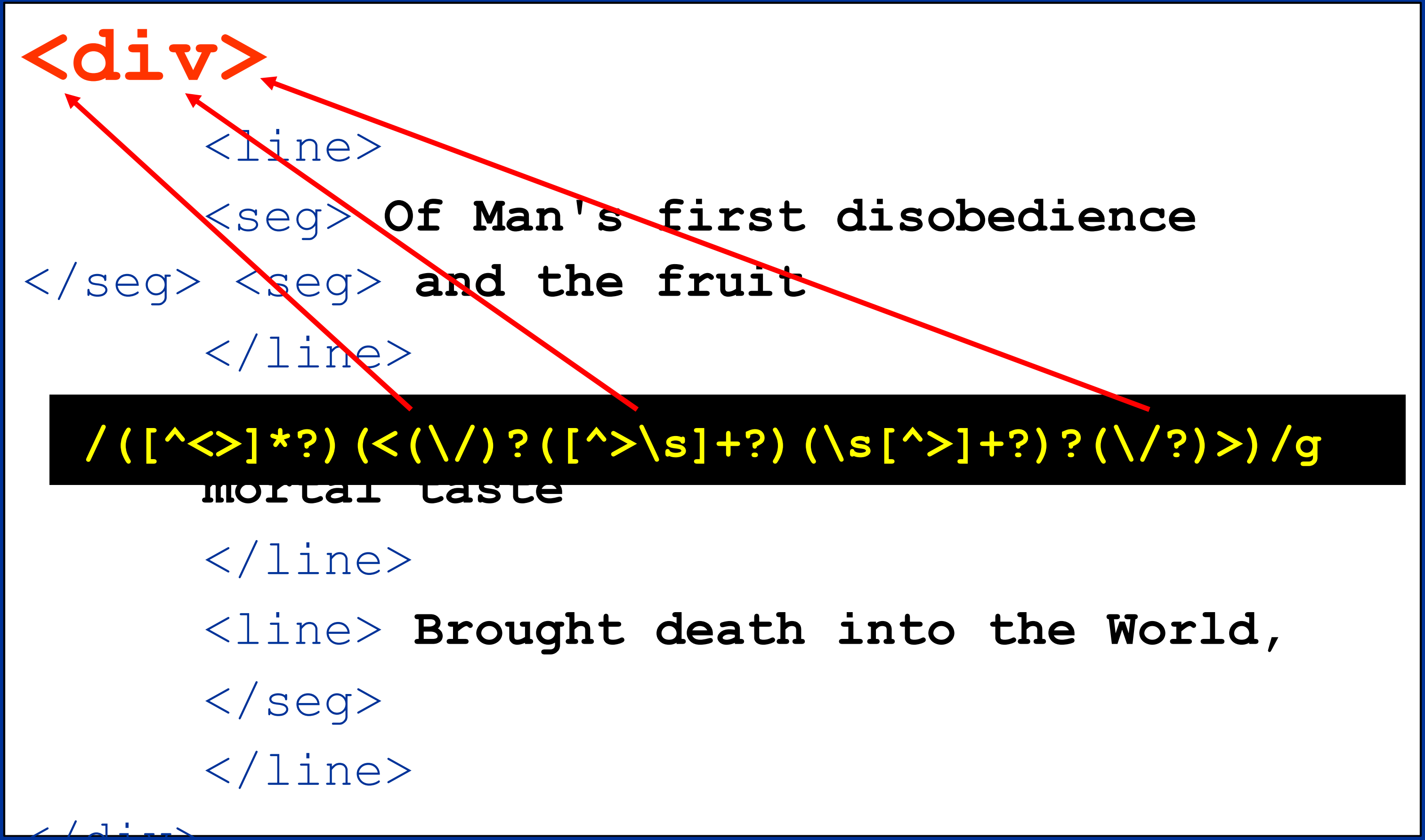
Input

```
<div>
      <line>
      <seg> Of Man's first disobedience
</seg> <seg> and the fruit
      </line>
      <line> Of that forbidden tree whose
      mortal taste
      </line>
      <line> Brought death into the World,
      </seg>
      </line>
</div>
```

jitts.org

# JITTs Filter Process

## morphological recognition



```
<div>
        <line>
        <seg> Of Man's first disobedience
</seg> <seg> and the fruit
        </line>
        mortal taste
        </line>
        <line> Brought death into the World,
        </seg>
        </line>
</div>
```

```
/([^<>]*?)(<(\/)?([^>\s]+?)(\s[^>]+?)?(\/?)>)/g
```

jitts.org

# JITTs Filter Process

vocabulary recognition

```
<div>
        <line>

</seg>

  /([^                                        >)/g



        <line> Brought death into the World,
        </seg>
        </line>
</div>
```

**Recognize:**

div

line

**Event:**

startElement('div')

jitts.org

# JITTs Filter Process

## morphological recognition

```
<div>
      <line>
      <seg> Of Man's first disobedience
</seg> <seg> and the fruit
      </line>
[illegible]
/([^<>]*?)(<(\/)?([^>\s]+?)(\s[^>]+?)?(\/?)>)/g
      mortal taste
      </line>
      <line> Brought death into the World,
      </seg>
      </line>
</div>
```

# JITTs Filter Process

vocabulary recognition

```
<div>
        <line>
</seg>
```

**Recognize:**

```
div
line
```

**Event:**

**startElement('line')**

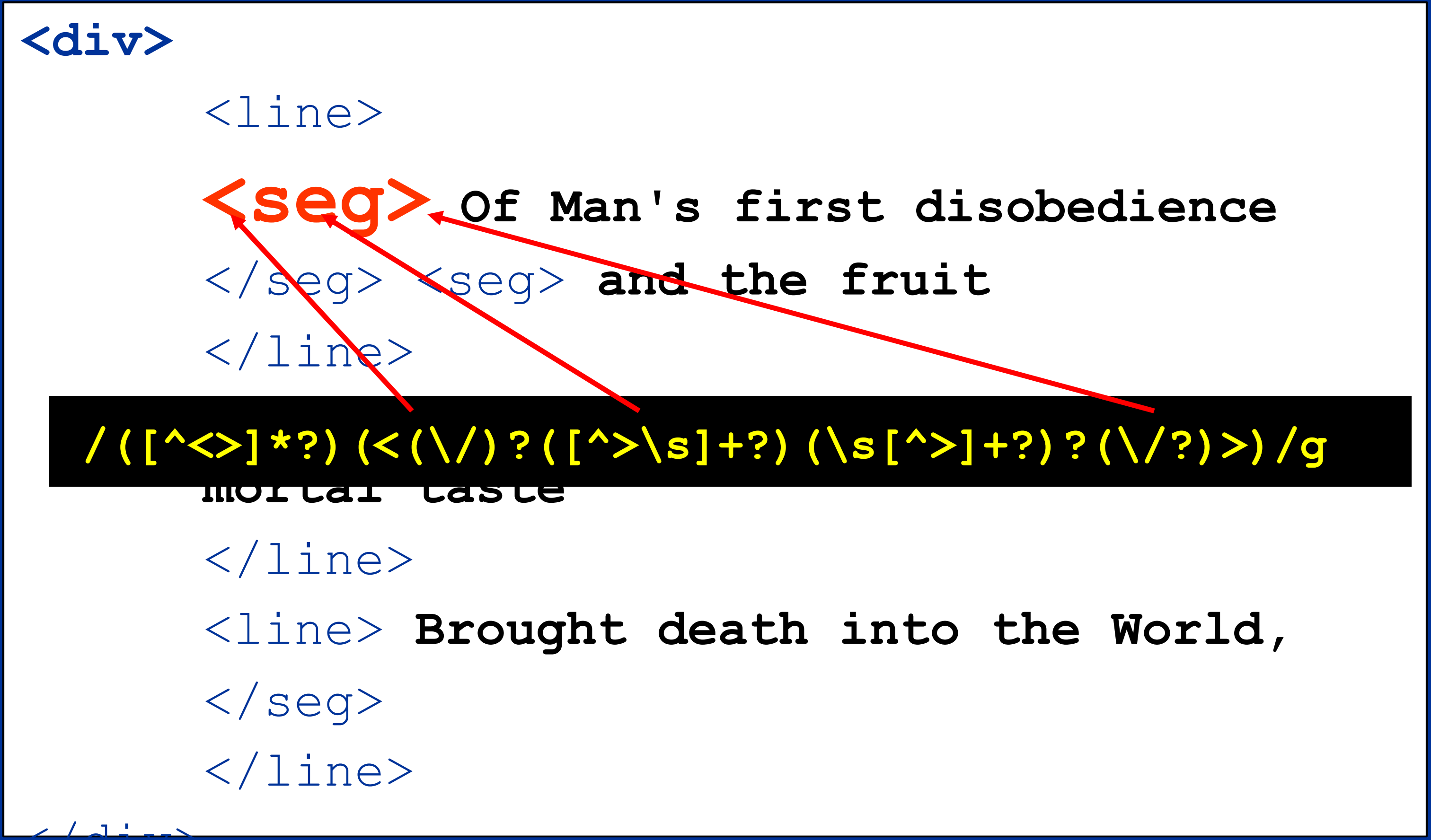
```
/([^ ... >)/g
```

```
        <line> Brought death into the World,
        </seg>
        </line>
```

jitts.org

# JITTs Filter Process

## morphological recognition



```
<div>
        <line>
        <seg> Of Man's first disobedience
        </seg> <seg> and the fruit
        </line>
/([^<>]*?)(<(\/)?([^>\s]+?)(\s[^>]+?)?(\/?)>)/g
        mortal taste
        </line>
        <line> Brought death into the World,
        </seg>
        </line>
</div>
```

jitts.org

# JITTs Filter Process

vocabulary recognition

**Recognize:**

div

line

```
<div>
      <line>
      <seg> Of Man's first disobedience
</seg> <seg> and the fruit
```

```
/([^                                  >)/g
```

**Event:**

*Either:* NO EVENT

*or:* characters('<seg>')

```
                                      ld,
      </seg>
      </line>
</div>
```

jitts.org

# JITTs Filter Process

## morphological recognition

```
<div>
      <line>
      <seg> Of Man's first disobedience
</seg> <seg> and the fruit
      </line>
 /([^<>]*?)(<(\/)?([^>\s]+?)(\s[^>]+?)?(\/?)>)/g
      mortal taste
      </line>
      <line> Brought death into the World,
      </seg>
      </line>
</div>
```

# JITTs Filter Process

## morphological recognition

```
<div>
      <line>
      <seg> Of Man's first disobedience
</seg> <seg> and the fruit
      </line>
 /([^                                          >)/g
                                              ld,
      </line>
</div>
```

**Event:**

characters(“Of Man’s first disobedience”)

# JITTs Filter Process

vocabulary recognition

**Recognize:**

div

line

```
<div>
      <line>
      <seg> Of Man's first
</seg> <seg> and the fruit
```

```
/([^                                  >)/g
```

**Event:**

*Either:* NO EVENT

*or:* characters('</seg>')

```
                                   ld,
      </line>
</div>
```

jitts.org

# JITTs Filter Process

## morphological recognition

```
<div>
        <line>
        <seg> Of Man's first disobedience
</seg> <seg> and the fruit
        </line>

 /([^<>]*?)(<(\/)?([^>\s]+?)(\s[^>]+?)?(\/?)>)/g
        mortal taste
        </line>
        <line> Brought death into the World,
        </seg>
        </line>
</div>
```

jitts.org

# JITTs Filter Process

## vocabulary recognition

**Recognize:**

div

line

```
<div>
      <line>
      <seg> Of Man's first
</seg> <seg> and the fruit
      </line>
 /([^                                    >)/g
                                          ld,
</div>
```

**Event:**

endElement('line')

jitts.org

# Overlapping Markup

```
<text>
<body>
<div>
<page><p>...</p>....</page>
<page><p>...</p>....</page>
<page><p>...</p></div><div>
<p>...</p>....</page>/div>
</body>
</text>
```

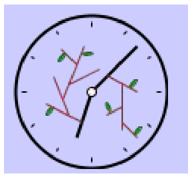
# Arbitrary tree

```
<text>
<body>

<page><p>...</p>....</page>
<page><p>...</p>....</page>
<page><p>...</p>
<p>...</p>..</page>
<p>...</p>..              </div>
</body>
</text>
```

```
<text>
<body>
<div>
        <p>...</p>....
        <p>...</p>....
        <p>...</p></div><div>


</body>
</text>
```

# Arbitrary CDATA

- CDATA indicates where "markup" not processed
- CDATA notation required due to conflating lexical/syntactical levels of markup language
- Separate lexical/syntactical levels, lose CDATA notation
- Declare element/children of element should be treated as CDATA



## What makes JITTs different?

- can already filter unwanted elements (SAX)
- can already extract arbitrary trees (XSLT, SAX)
- JITTs can validate alternative trees (inline)
- JITTs can preserve arbitrary markup for later processing
- JITTs corrects lexical/syntactical conflation of SGML/XML

jitts.org

# Schrödinger's Document

```
<?xml version="1.0" encoding="UTF-8"?>
```

a. Is the document inside valid or not?

a. Can't determine without processing the document

a. What does processing mean? Recognizing a tree in the document!

## Advantages of JITTs

- enforced or Just-In-Time-Validation (cf. conference submissions, partial texts)
- multiple views of a text
- trees are imposed on a text (cf. DATATAG, markup as well)
- preservation of unrecognized markup = DOM Lite!

## DOM-Lite

- DOM trees built upon markup events
- no element recognized = no event = no DOM node
- Example: dictionary entries complex internally but most access by headword
- If headword is node in DOM tree, stream out internal markup for presentation (no lower nodes in the DOM tree)

## What’s different in JITTs parsing?

a. element recognition is no longer "<" and "</"

a. element recognition is "<" plus element name, "</" plus element name

a. building tree for the DTD or Schema anyway, just using it differently

## What’s different in JITTs parsing?

a. tree built for the document only uses the recognized markup

a. markup that is not recognized is passed as PCDATA

a. or discarded, depending upon the user's needs

jitts.org

## Objection!
## JITTS supports invalid SGML/XML

a. Recall Schrödinger's Document: a document is neither valid nor invalid until processed

a. can opt for traditional validation or can use JITTs

# OK, put that statement to a customer who needs:

a. versioning

a. JITV (Just-in-time-validation)

a. DOM Lite

a. Reply to the objection is:

Can you say **NO SALE**?

## Beyond JITTs

a. how to recover the membership insight from BUVH?

a. most overlap is localized so find the string and parse backwards until trees converge

# Beyond JITTs

a. still have problems with discontinuous segments (overlap+)

a. looking at use of set processing to further manipulate XML encoded texts

## Observation #4

**Key to progress is not being locked into a single processing view for XML**

## What does “Tree” mean and when? (Apologies to Lewis Carroll)

"When I use 'tree'," Patrick said in a scornful tone, "it means just what I choose it to mean--neither more nor less."

"The question is," said the W3C, "whether you can make 'tree' mean so many different things."

"The question is," said Patrick, "which is to be master--that's all."

## Conclusions

JITTs requires minimal change in parsing of markup

JITTs retains validation of markup

JITTs corrects conflation of lexical vs. syntactical levels in markup

JITTs enables arbitrarily complex markup

JITTs enables DOM (Lite) for large texts

JITTs changes processing, not syntax

jitts.org

## Support for Research

- Support organizations that make this research possible!
  - Society of Biblical Literature: http://www.sbl-site.org
  - OpenText.org: http://www.opentext.org
  - TEI: http://www.tei-c.org