

# XML Syntax and Editors/Validator

<https://www.w3schools.com/xml/>

Keywords: Syntax, Editor – XMLViewer, XML Validator

<https://codebeautify.org/xmlviewer#>

## XML Syntax Checking; XML Node Hierarchy

The screenshot shows the Code Beautify XML Viewer web application. The browser's address bar displays the URL `codebeautify.org/xmlviewer#`. The page header includes the Code Beautify logo and navigation links for JSON Formatter, XML Formatter, Calculators, JSON Beautifier, Recent Links, Sitemap, and a Login button. Below the header, the title "XML Viewer" is prominently displayed. A yellow banner on the right side of the page contains the text "XML Syntax Checking; XML Node Hierarchy". A central banner for Adobe promotes a 15% discount on 20+ apps. The main interface is divided into three sections: a left pane for inputting XML data (labeled "Sample" with a "Paste or type your data here...." prompt), a central control pane with buttons for "File", "URL", "Auto Update", "Tree View", "Web Content Data Entry", "Beautify", "Minify", "to JSON", "to CSV", and "Download", and a right pane for the "Output". The status bar at the bottom of both the input and output panes shows "Ln: 1 Col: 0".

# <https://codebeautify.org/xmlviewer#>

The screenshot displays the CodeBeautify XML Viewer interface, which is used for visualizing and manipulating XML data. The interface is divided into several sections:

- Input Section:** Contains a text area for pasting XML code. The sample XML provided is:

```
1 <?xml version="1.0" encoding="euc-kr" standalone="yes" ?>
2 <?xml-stylesheet type="text/css" href="memo1_Simplified2.css"?>
3 <MEMO>
4   <TO> To: &quot; Eugene &quot;</TO>
5   <FROM> From: Hong, Gil-Dong</FROM>
6   <CONTENTS> Can we dine out at 1 PM?</CONTENTS>
7 </MEMO>
```
- File/URL Section:** Includes buttons for 'File' and 'URL', and a checkbox for 'Auto Update'.
- Tree View:** A button labeled 'Tree View' (highlighted with a blue arrow) that toggles the display of the XML data as a hierarchical tree structure.
- Web Content Data Entry:** A section for automating web content data entry with AI, featuring an 'OPEN' button.
- Beautify/Minify Section:** Includes buttons for 'Beautify' and 'Minify', and a 'Download' button.
- Output Section:** Displays the result of the operations. It includes:
  - Tree View Output:** A hierarchical representation of the XML data, showing the root 'memo' and its children 'to', 'from', and 'contents' (highlighted with a blue arrow).
  - JSON Output:** A text area showing the XML data converted to JSON format (highlighted with a blue arrow). The JSON output is:

```
1 {
2   "MEMO": {
3     "TO": "To: &quot; Eugene &quot;",
4     "FROM": "From: Hong, Gil-Dong",
5     "CONTENTS": "Can we dine out at 1 PM?"
6   }
7 }
```
  - CSV Output:** A text area showing the XML data converted to CSV format. The CSV output is:

```
1 MEMO/TO,MEMO/FROM,MEMO/CONTENTS
2 "To: "" Eugene ""","From: Hong, Gil-Dong","Can we
3   out at 1 PM?"
```

<https://codebeautify.org/xmlviewer>

The screenshot displays the Code Beautify XML Viewer interface. On the left, the 'Input' XML is shown with line numbers 1 through 10. The XML content is:

```
1 <?xml version="1.0" encoding="euc-kr" standalone="yes" ?>
2 <?xml-stylesheet type="text/css" href="memo1_Simplified2.css"?>
3 <MEMO>
4   <TO> To: &quot; Eugene &quot;</TO>
5   <FROM> From: Hong, Gil-Dong</FROM>
6   <CONTENTS> Can we dine out at 1 PM?</CONTENTS>
7 </MEMO>
8
9
10
```

In the center, there are controls for 'File' and 'URL' input, an 'Auto Update' checkbox (checked), and a 'Tree View' button. Below these is a 'Web Content Data Entry' section with a description and a 'EN' button. At the bottom, there are 'Beautify' and 'Minify' buttons. Blue arrows point from the 'EN' button to both 'Beautify' and 'Minify'.

On the right, the 'Output' window shows the XML after the 'Beautify' operation, with line numbers 1 through 8:

```
1 <?xml version="1.0" encoding="euc-kr" standalone="yes" ?>
2 <?xml-stylesheet type="text/css" href="memo1_Simplified2.css"?>
3 <MEMO>
4   <TO> To: &quot; Eugene &quot;</TO>
5   <FROM> From: Hong, Gil-Dong</FROM>
6   <CONTENTS> Can we dine out at 1 PM?</CONTENTS>
7 </MEMO>
8
```

This screenshot shows the 'Output' window after the 'Minify' operation. The XML is now compacted into a single line, with line numbers 1 and 2 visible:

```
1 <?xml version="1.0" encoding="euc-kr" standalone="yes" ?><?xml-stylesheet type="text/css" href="memo1_Simplified2.css"?><MEMO><TO> To: &quot; Eugene &quot;</TO><FROM> From: Hong, Gil-Dong</FROM><CONTENTS> Can we dine out at 1 PM?</CONTENTS></MEMO>
2
```

<https://codebeautify.org/xmlviewer>

The image shows a screenshot of the XML Viewer tool interface, which is divided into three main sections: a code editor, a central control panel, and an output panel.

**Code Editor (Left):** Displays the XML document. The code is as follows:

```
1 <?xml version="1.0" encoding="euc-kr" standalone="yes" ?>
2 <?xml-stylesheet type="text/css" href="memo1_Simplified2.css"?>
3 <MEMO>
4   <TO> To: &quot; Eugene &quot;</TO>
5   <FROM> From: Hong, Gil-Dong</FROM>
6   <CONTENTS> Can we dine out at 1 PM?</CONTENTS>
7 </MEMO1>
8
9
10
```

A red arrow points to the closing tag `</MEMO1>` on line 7, indicating a mismatch with the opening tag `<MEMO>` on line 3.

**Central Control Panel:** Contains options for file loading (File, URL), an "Auto Update" checkbox, and a "Tree View" button. Below these is a preview of the XML content, which is a "Web Content Data Entry" form. At the bottom of this panel are buttons for "Beautify", "Minify", "to JSON", and "to CSV".

**Output Panel (Right):** Displays the parsed XML structure and any errors. It shows a tree view where the root element is `memo`, which contains a `parsererror` element. The error message states: "error on line 7 at column 9: Opening and ending tag mismatch: MEMO line 3 and MEMO1". A red arrow points to this error message. Below the error, the tool provides a rendering of the XML content up to the first error, showing the "To: Eugene", "From: Hong, Gil-Dong", and "contents: Can we dine out at 1 PM?" fields.

# HTML Viewer

<https://codebeautify.org/htmlviewer>

The screenshot shows the Code Beautify HTML Viewer interface. The browser address bar displays `codebeautify.org/htmlviewer#`. The site's navigation bar includes links for JSON Formatter, XML Formatter, Calculators, JSON Beautifier, Recent Links, and Sitemap. The main heading is "HTML Viewer". A yellow warning box states: "warning: missing ending tags of em and p". A blue button labeled "Try to fix syntax errors." is visible. The code editor on the left contains the following HTML code:

```
<HTML>
<HEAD>
<TITLE>MEMO</TITLE>
</HEAD>
<BODY>
<P style = "font-weight:bold">To : Eugene
<P style = "font-weight:bold">From : Hong, Gil-Dong
<P style = "font-weight:bold; color: red"><em>Can we
    meet tomorrow?<em>
</BODY>
</HTML>
```

Lines 9 and 10 of the code are marked with red 'x' icons, indicating syntax errors. A red arrow points to the error icon on line 10. The right panel shows the "Output" preview, which displays the rendered HTML:

To : Eugene  
From : Hong, Gil-Dong  
*Can we meet tomorrow?*

# HTML Viewer



의견을 공유해주셔서 감사합니다.

Ad choices ▶



Sample



```
i 1 <HTML>
  2 <HEAD>
  3 <TITLE>MEMO</TITLE>
  4 </HEAD>
  5 <BODY>
  6 <P style = "font-weight:bold">To : Eugene </p>
  7 <P style = "font-weight:bold">From : Hong, Gil-Dong
    </p>
  8 <P style = "font-weight:bold; color: red"><em>Can we
    meet tomorrow?</em> </p>
  9 </BODY>
 10 </HTML>
```

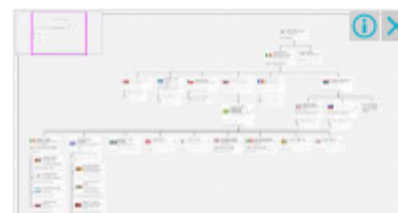
File

URL

☒ Auto  
Update

2 Space ▼

Beautify HTML



Output

To : Eugene

From : Hong, Gil-Dong

*Can we meet tomorrow?*

<https://codebeautify.org/htmlviewer>

The screenshot displays the Code Beautify HTML Viewer interface. The main editor on the left contains the following HTML code:

```
1 <HTML>
2 <HEAD>
3 <TITLE>MEMO</TITLE>
4 </HEAD>
5 <BODY>
6 <P style = "font-weight:bold">To : Eugene </p>
7 <P style = "font-weight:bold">From : Hong, Gil-Dong
  </p>
8 <P style = "font-weight:bold; color: red"><em>Can we
  meet tomorrow?</em> </p>
9 </BODY>
10 </HTML>
```

The right sidebar contains the following controls:

- Buttons: File, URL
- Checkbox: ☒ Auto Update
- Dropdown: 2 Space
- Button: Beautify HTML
- Output panel showing the beautified HTML:

```
1 <html><head><title>MEMO</title></head><body><p style
  ="font-weight:700">To : Eugene</p><p style="font
  -weight:700">From : Hong, Gil-Dong</p><p style
  ="font-weight:700;color:red"><em>Can we meet
  tomorrow?</em></p></body></html>
```

At the bottom of the sidebar, there are buttons for Run / View and Minify HTML. A blue arrow points to the Minify HTML button.



# HTML Viewer

```
Sample [refresh] [folder] [save] [trash] [copy] [share]
1 <HTML>
2 <HEAD>
3 <TITLE>MEMO</TITLE>
4 </HEAD>
5 <BODY>
6 <P style = "font-weight:bold">To : Eugene
7 <P style = "font-weight:bold">From : Hong, Gil-Dong
8 <P style = "font-weight:bold; color: red"><em>Can we
  meet tomorrow?
9 </BODY>
10 </HTML>
11
```

File URL

☒ Auto Update 2 Space

Beautify HTML

Run / View

Minify HTML

Output

To : Eugene

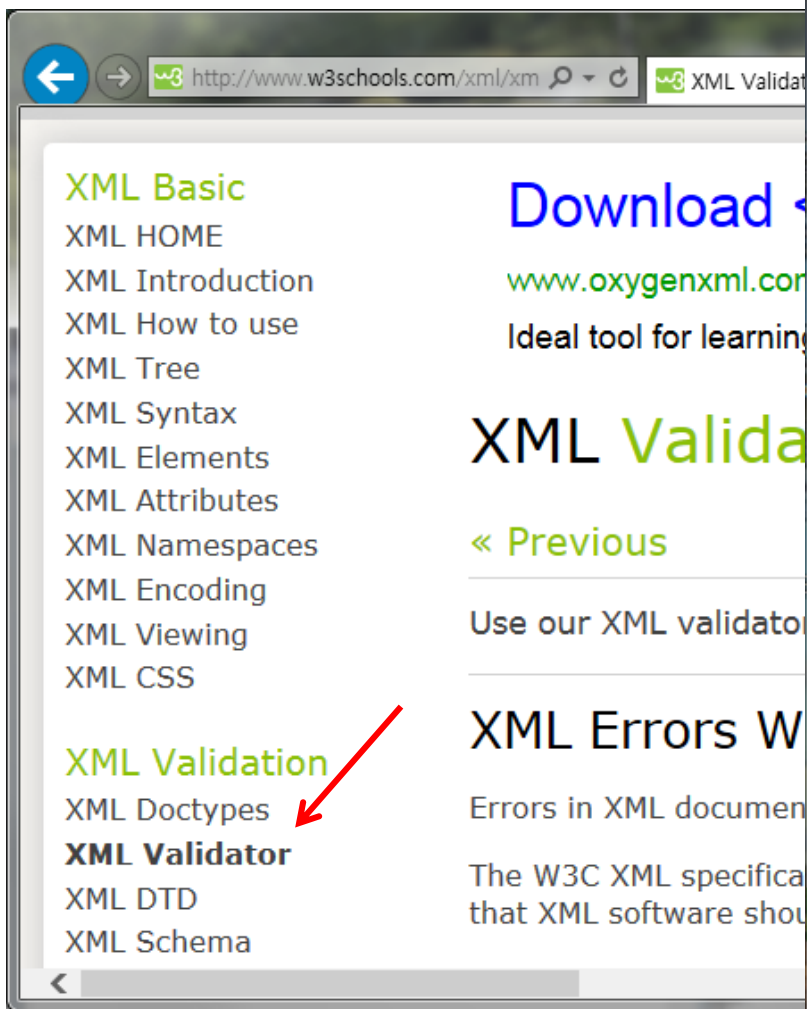
From : Hong, Gil-Dong

*Can we meet tomorrow?*

```
Sample [refresh] [folder] [save] [trash] [copy] [share]
1 <HTML>
2 <HEAD>
3 <TITLE>MEMO</TITLE>
4 </HEAD>
5 <BODY>
6 <P style = "font-weight:bold">To : Eugene
7 <P style = "font-weight:bold">From : Hong, Gil-Dong
8 <P style = "font-weight:bold; color: red"><em>Can we
  meet tomorrow?</em></p>
9 </BODY>
10 </HTML>
11
```

warning: missing ending tags  
of em and p

[http://www.w3schools.com/xml/xml\\_validator.asp](http://www.w3schools.com/xml/xml_validator.asp)



[Home](#) [HTML](#) [CSS](#) [JAVASCRIPT](#) [SQL](#) [PYTHON](#) [MORE ▾](#)

- XML Introduction
- XML How to use
- XML Tree
- XML Syntax
- XML Elements
- XML Attributes
- XML Namespaces
- XML Display
- XML HttpRequest
- XML Parser
- XML DOM
- XML XPath
- XML XSLT
- XML XQuery
- XML XLink
- XML Validator**
- XML DTD
- XML Schema
- XML Server
- XML Examples
- XML Quiz
- XML Certificate

To help you syntax-check your XML, we have created an

Try to syntax-check correct XML :

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Try to syntax-check incorrect XML :

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</pheading>
<body>Don't forget me this weekend!</body>
</note>
```

Checking Syntax

# Try using XML with W3Schools Validator.

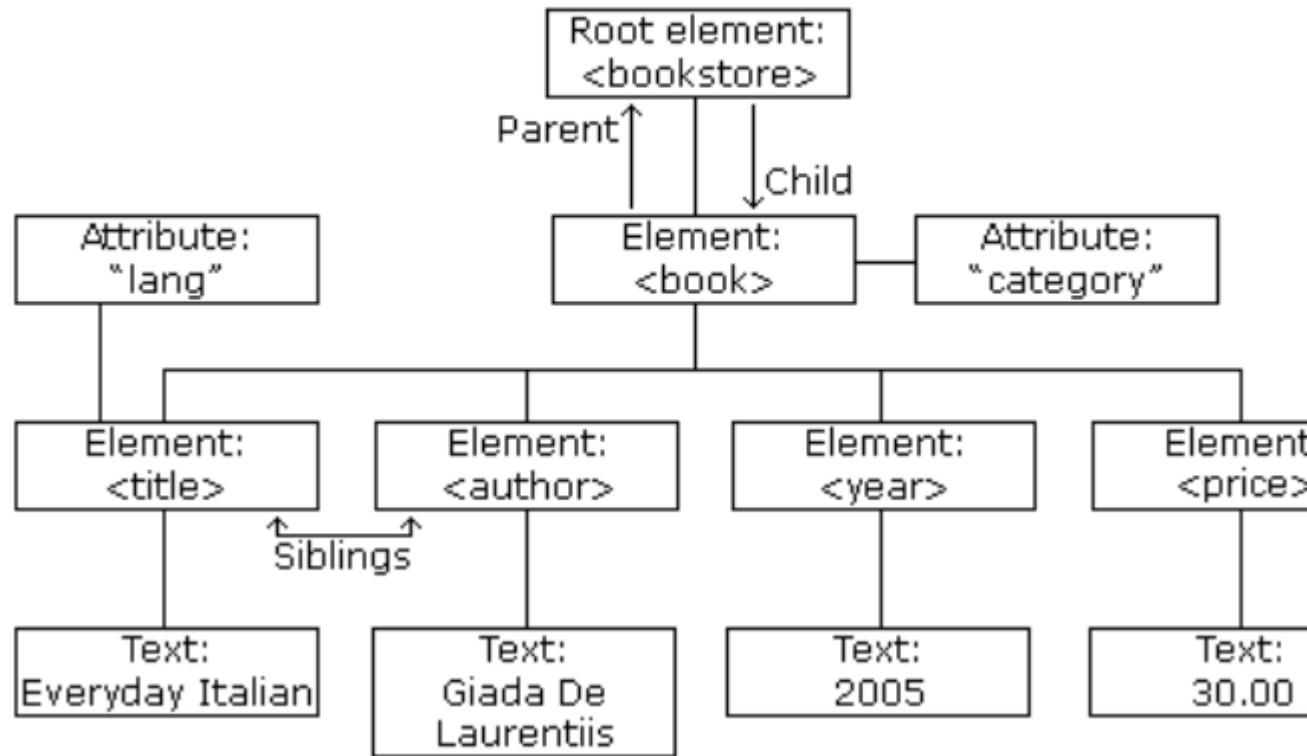
[https://www.w3schools.com/xml/xml\\_validator.asp](https://www.w3schools.com/xml/xml_validator.asp)

Try to syntax-check incorrect XML :

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</pheading>
<body>Don't forget me this weekend!</body>
</note>
```

Try to syntax-check your own XML :

# XML Tree Structure



```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

## XML Viewer

❤ Add to Fav

New

Save &amp; Share

```
Sample ↻ 📁 💾 ✓ 🖨 🗑 📄 🔗
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/css" href="books_original.css"
   ?>
3 <bookstore>
4
5   <book category="cooking">
6     <title lang="en">Everyday Italian</title>
7     <author>Giada De Laurentiis</author>
8     <year>2005</year>
9     <price>30.00</price>
10  </book>
11
12  <book category="children">
13    <title lang="en">Harry Potter</title>
14    <author>J K. Rowling</author>
15    <year>2005</year>
16    <price>29.99</price>
17  </book>
18
19  <book category="web">
20    <title lang="en">XQuery Kick Start</title>
21    <author>James McGovern</author>
22    <author>Per Bothner</author>
23    <author>Kurt Cagle</author>
24    <author>James Linn</author>
25    <author>Vaidyanathan Nagarajan</author>
26    <year>2003</year>
```

Ln: 37 Col: 12 size: 1.02 KB T T

📁 File

🔗 URL

☒ Auto Update

Tree View

Beautify

Minify

to JSON

to CSV

📄 Download

Output

```
bookstore ..
├── book ..
│   ├── @category: cooking
│   ├── title Everyday Italian
│   │   └── @lang: en
│   ├── author Giada De Laurentiis
│   ├── year 2005
│   └── price 30.00
└── book ..
    ├── @category: children
    ├── title Harry Potter
    │   └── @lang: en
    └── author J K. Rowling
```

# Code Beautify shows a tree structure...

The screenshot displays the Code Beautify web application interface, which is divided into three main sections:

- Code Editor:** The leftmost section shows the XML code being processed. The code is as follows:

```
1 <?xml version="1.0" encoding="euc-kr" standalone="yes"?>
2 <?xml-stylesheet type="text/css" href="memo1_Simplified2.css"?>
3 <MEMO>
4   <TO> To: " Eugene ";</TO>
5   <FROM> From: Hong, Gil-Dong</FROM>
6   <CONTENTS> Can we dine out at 1 PM?</CONTENTS>
7 </MEMO>
8
9
10
```
- Controls:** The middle section contains a toolbar with buttons for File and URL, an "Auto Update" checkbox (which is checked), and a prominent green "Tree View" button. Below these are two buttons: "Ad was inappropriate" and "Not interested in this ad".
- Output:** The rightmost section, titled "Output", displays the resulting DOM tree structure. The tree is visualized with nodes and connecting lines:
  - The root node is **memo**.
  - Under **memo**, there are three child nodes: **to**, **from**, and **contents**.
  - The **to** node has the text "To: " Eugene " " as its value.
  - The **from** node has the text "From: Hong, Gil-Dong" as its value.
  - The **contents** node has the text "Can we dine out at 1 PM?" as its value.

# What is an XML Element?

An XML element is everything from (including) the element's start tag to (including) the element's end tag.

```
<price>29.99</price>
```

An element can contain:

- text
- attributes
- other elements
- or a mix of the above

```
<bookstore>
  <book category="children">
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

# Empty XML Elements

An element with no content is said to be empty.

In XML, you can indicate an empty element like this:

```
<element></element>
```

You can also use a so called self-closing tag:

```
<element />
```



# Syntax: XML Naming Rules

- Element names are **case-sensitive**
- Element names must start with a **letter** or **underscore**
- Element names cannot start with the letters xml (or XML, or Xml, etc) – **not anymore, but discouraged**
- Element names can contain letters, digits, **hyphens, underscores, and periods**
- Element names cannot contain spaces

# Exercise XML syntax with "Code Beautify".

The screenshot displays the Code Beautify web application interface. On the left, the XML source code is shown with line numbers 1 through 12. The code includes an XML declaration, a stylesheet reference, and a root element named 'MEMO' containing 'TO', 'xml', 'memo', 'FROM', and 'CONTENTS' elements. The 'memo' element is currently selected. In the center, there are controls for 'File' and 'URL' input, an 'Auto Update' checkbox, and a 'Tree View' button. Below these is a preview area showing a 'Springer' logo and the text 'Preparing your manuscript?'. On the right, the 'Output' panel shows a hierarchical tree view of the XML document, with nodes labeled 'memo', 'to', 'xml', 'memo', 'from', and 'contents', each followed by its corresponding text content.

```
1 <?xml version="1.0" encoding="euc-kr" standalone="yes" ?>
2 <?xml-stylesheet type="text/css" href="memo1_Simplified2.css"?>
3 <MEMO>
4   <TO> To: " Eugene " </TO>
5   <xml> xml contents </xml>
6   <memo> memo contents </memo>
7   <FROM> From: Hong, Gil-Dong</FROM>
8   <CONTENTS> Can we dine out at 1 PM?</CONTENTS>
9 </MEMO>
10
11
12
```

File URL

☒ Auto Update

Tree View

Springer

Preparing your manuscript?

Output

- memo ..
  - to To: " Eugene "
  - xml xml contents
  - memo memo contents
  - from From: Hong, Gil-Dong
  - contents Can we dine out at 1 PM?

automatically making ending tag...

# JSON Formatter | Hex Color Codes | HTML Generator | my ip | Search | Recent Links | mo

## XML Viewer☆

### XML Input

 [sample](#)  

```
1 <?xml version="1.0" encoding="euc-kr"
   standa one="yes"?>
2 <?xml-stylesheet type="text/css" href
   = "emo1.css"?>
3 <xml>
4   <TO> To: Eugene</TO>
5   <FROM> From: Hong, Gil-Dong</FROM>
6   <CONTENTS> Can we dine out at 1 PM
   ?</CONTENTS>
7 </xml>
8
```

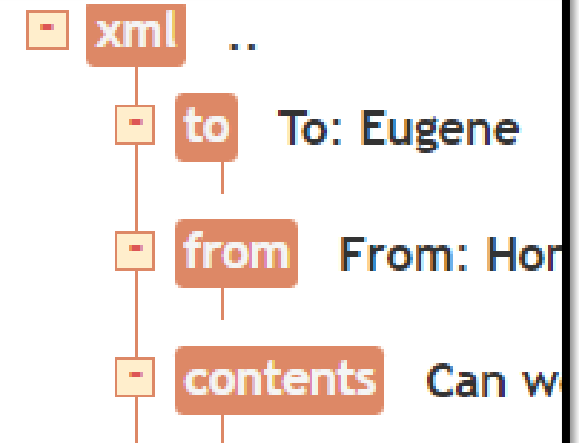
Load Url

Browse

Tree View

Beautify / Form

### Result : XML Tree



Sample ↺ 📁 💾 ✓ 🖨️ 🗑️ 📄 ↗️

1 <?xml version="1.0" encoding="euc-kr" standalone="yes" ?>

2 <?xml-stylesheet type="text/css" href="memo1\_Simplified2.css"?>

3 <MEMO>

4 <TO> To: " Eugene "

5 <xml> xml contents </xml>

6 <xml />

7 <memo> memo contents </memo>

8 <FROM> From: Hong, Gil-Dong</FROM>

9 <CONTENTS> Can we dine out at 1 PM?</CONTENTS>

10 </MEMO>

11

12

13

📁 File 🔗 URL

☒ Auto Update

Tree View

🗣️ 챗봇 챗지피티

AI의 모든 기능을 갖추고 있습니다. 채팅, 글쓰기, 번역, 요약, 이미지 생성 등을 포함합니다.

Monica AI

열기

Output

memo ..

to To: " Eugene "

xml xml contents

xml

memo memo contents

from From: Hong, Gil-Dong

contents Can we dine out at 1 PM?

# Syntax: Best Naming Practices

- Create **descriptive** names, like this: <person>, <firstname>, <lastname>.
- Create **short and simple** names, like this: <book\_title> not like this: <the\_title\_of\_the\_book>.
- **Avoid "-"**. If you name something "first-name", some software may think you want to **subtract** "name" from "first".
- **Avoid "."**. If you name something "first.name", some software may think that "name" is a property of the object "first".

# Naming Styles

no naming styles defined for XML elements

Style	Example	Description
Lower case	<code>&lt;firstname&gt;</code>	All letters lower case
Upper case	<code>&lt;FIRSTNAME&gt;</code>	All letters upper case
Underscore	<code>&lt;first_name&gt;</code>	Underscore separates words
Pascal case	<code>&lt;FirstName&gt;</code>	Uppercase first letter in each word
Camel case	<code>&lt;firstName&gt;</code>	Uppercase first letter in each word except the first

Camel case is a common naming rule in JavaScripts.

# XML Tags are Case Sensitive

XML tags are case sensitive. The tag <Letter> is different from the tag <letter>.

Opening and closing tags must be written with the same case:

```
<message>This is correct</message>
```

## XML Attribute Values Must Always be Quoted

XML elements can have attributes in name/value pairs just like in HTML.

In XML, the attribute values must always be quoted:

```
<note date="12/11/2007">  
  <to>Tove</to>  
  <from>Jani</from>  
</note>
```

# XML Elements Must be Properly Nested

In HTML, you might see improperly nested elements:

```
<b><i>This text is bold and italic</b></i>
```

In XML, all elements **must** be properly nested within each other:

```
<b><i>This text is bold and italic</i></b>
```



# tag overlapping

Type the source for HTML and XML, then apply them to the browser.

```
<HTML>
<HEAD>
<TITLE>Tag Rule</TITLE>
</HEAD>
<BODY>
<B>bold face <I>bold italic </B>
italic</I>
</BODY>
</HTML>
```

```
<?xml version="1.0" encoding="euc-kr" ?>
<tag>
<B>bold face <I>bold italic </B>italic </I>
</tag>
```

