XML Formatting Cheatsheet

A comprehensive guide to XML formatting, covering syntax, best practices, and common formatting techniques for creating readable and maintainable XML documents.

XML Structure and Syntax

Basic XML Structure

XML documents must have a root element that contains all other elements.

Example:

```
<root>
  <element>Content</element>
</root>
```

XML elements consist of a start tag, content, and an end tag.

Example:

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

Elements can have attributes that provide additional information.

Example:

```
<element
attribute="value">Content</element>
```

XML Declaration

The XML declaration is optional but recommended. It specifies the XML version and encoding.

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

Version attribute	Specifies the XML version being used (usually 1.0).
Encoding attribute	Specifies the character encoding (e.g., UTF-8, ISO-8859-1).

Comments

Comments are used to include explanatory notes in the XML document.

Example:

```
<!-- This is a comment -->
```

Comments can span multiple lines.

Example:

```
<!--
  This is a multi-line
  comment.
-->
```

Formatting Best Practices

Indentation

Use consistent indentation to improve readability. Common indentation is 2 or 4 spaces.

Example:

```
<root>
  <element>
    <subelement>Content</subelement>
  </element>
</root>
```

Avoid using tabs for indentation, as they may be displayed differently in different editors.

Line Breaks

Add line breaks after each start and end tag to enhance readability, especially for complex structures.

```
<root>

<elemen
t>

Content

</eleme
nt>
</root>
```

For elements with only text content, a single line is acceptable.

```
<elemen
t>Conte
nt</ele
ment>
```

Attribute Formatting

Place each attribute on a new line if there are multiple attributes to improve readability.

Example:

```
<element
  attribute1="value1"
  attribute2="value2">
  Content
</element>
```

Ensure attribute values are properly quoted (using either single or double quotes).

Handling Special Characters and CDATA

Escaping Special Characters

Special characters in XML must be escaped using predefined entities.

Example:

- `<` (less than) becomes `<`
- `>` (greater than) becomes `>`
- `&` (ampersand) becomes `&`
- `'` (apostrophe) becomes `&apos;`
- `"` (double quote) becomes `&quot;`

Use these entities within element content and attribute values to avoid parsing errors.

CDATA Sections

CDATA sections are used to include blocks of text that contain special characters without escaping them.

```
<![CDATA[

<element>Content with < and > characters</element>
]]>
```

CDATA sections start with `<![CDATA[` and end with `]]>`.

Within a CDATA section, only `]]>` is recognized as a special sequence.

Whitespace Handling

XML processors preserve whitespace by default. Significant whitespace should be handled carefully.

Example:

```
<root>
  <element>  Content  </element>
</root>
```

Use the `xml:space` attribute to control whitespace handling if needed. The value can be `default` or `preserve`.

Example:

```
<element xml:space="preserve">  Content
</element>
```

Advanced Formatting Techniques

Using XML Schema for Validation

XML Schema Definition (XSD) can be used to validate the structure and content of XML documents.

Example:

```
<root
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="namespace schema.xsd">
  ...
</root>
```

Validating XML against a schema ensures consistency and correctness.

Pretty Printing

Pretty printing involves automatically formatting XML with indentation and line breaks for better readability.

Many XML editors and libraries provide pretty printing functionality.

Tools like `xmllint` can be used for command-line pretty printing.

```
xmllint --format --indent 4 input.xml
```

Namespace Management

XML namespaces provide a way to avoid naming conflicts between elements and attributes from different sources.

Example:

```
<root
xmlns:prefix="http://example.com/namespace">

<prefix:element>Content</prefix:element>
</root>
```

Use namespaces to organize and differentiate elements in complex XML documents.