**XML (eXtensible Markup Language) -** [**https://www.w3schools.com/xml/xml\_whatis.asp**](https://www.w3schools.com/xml/xml_whatis.asp)

È uno strumento indipendente dal software e dall'hardware per la memorizzazione e il trasporto dei dati.

<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>

L'XML è abbastanza ***auto-descrittivo***:

* Ha informazioni sul mittente.
* Ha informazioni sul ricevitore
* Ha un titolo
* Ha un corpo di messaggio.

XML e HTML sono stati progettati con diversi obiettivi:

* XML è stato progettato per il trasporto di dati, con particolare attenzione ai dati
* L'HTML è stato progettato per visualizzare i dati, con particolare attenzione all'aspetto dei dati
* *I tag XML non sono predefiniti* come i tag HTML

Con XML, l'autore deve definire sia i tag che la struttura del documento.

La maggior parte delle applicazioni XML funzionerà anche se vengono aggiunti (o rimossi) nuovi dati.

***XML separa i dati dalla presentazione,*** non contiene alcuna informazione su come vengono visualizzato, e possono essere utilizzati in molti diversi scenari di presentazione.

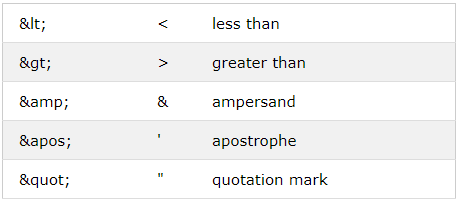
***XML separa i dati da HTML,*** quando si visualizzano i dati in HTML, non è necessario modificare il file HTML quando i dati cambiano. Con XML, i dati possono essere archiviati in file XML separati.

*Un documento XML si dice* ***ben formato*** *(****well-formed****) se sono rispettate le seguenti regole:*

* *Il documento XML contiene un unico elemento* ***root*** *(“radice” dell’albero);*
* *Gli elementi devono essere sempre chiusi con* ***tag di chiusura*** *o, se vuoti, tramite chiusura abbreviata (/>);*
* *Bisogna rispettare* ***l’ordine di nidificazione****: un elemento padre non può essere chiuso prima di un elemento figli;*
* *XML è* ***case sensitive****: bisogna ricordarlo quando usiamo maiuscole e minuscole per nomi dei tag e attributi*
* *Gli* ***attributi*** *devono essere racchiusi* ***tra singoli o doppi apici***

***Regole di sintassi -*** [***https://www.w3schools.com/xml/xml\_syntax.asp***](https://www.w3schools.com/xml/xml_syntax.asp)

* I documenti XML devono contenere un elemento ***radice*** che è il ***genitore*** di tutti gli altri elementi
* Il ***prologo*** XML è facoltativo. Se esiste, deve venire prima nel documento. Per evitare errori, è necessario specificare la codifica utilizzata o salvare i file XML come UTF-8 (codifica dei caratteri predefinita per i documenti XML).
* Tutti gli elementi ***devono*** avere un tag di chiusura.
* I tag XML sono ***case sensitive***. Il tag <Lettera> è diverso dal tag <lettera>.
* Gli elementi XML devono essere ***nidificati correttamente.***
* Gli elementi XML possono avere ***attributi*** in coppie nome / valore proprio come in HTML, e i suoi valori devono sempre essere citati.
* ***Riferimenti di entità***, es. sostituire il carattere "<" con un ***riferimento di entità***.

******

* La sintassi per scrivere ***commenti*** in XML è simile a quella dell'HTML: <!-- This is a comment -->
* *XML non tronca più* ***spazi bianchi****.*

Si dice che i documenti XML conformi alle regole di sintassi sopra siano documenti XML "***ben formati***".

***Albero XML -*** [***https://www.w3schools.com/xml/xml\_tree.asp***](https://www.w3schools.com/xml/xml_tree.asp)

I documenti XML sono formati come ***alberi di elementi***. Un albero XML inizia da un elemento radice coi suoi elementi figlio. Tutti gli elementi possono avere elementi secondari.

<root>  
  <child>  
    <subchild>.....</subchild>  
  </child>  
</root>

<?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 utilizza una sintassi molto auto-descrittiva:

* Un ***prolog*** definisce la versione XML e la codifica dei caratteri:

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

* La riga successiva è l'***elemento principale*** del documento:

<bookstore>

* La riga successiva inizia un elemento <book>:

<book category="cooking">

* Gli elementi <book> hanno 4 elementi figlio : <title>, <author>, <year>, <price>:

<title lang="en">Everyday Italian</title>  
<author>Giada De Laurentiis</author>  
<year>2005</year>  
<price>30.00</price>

***Elementi -*** <https://www.w3schools.com/xml/xml_elements.asp>

Un elemento XML è tutto da (incluso) il tag di inizio dell'elemento a (incluso) il tag di fine.

<price>29.99</price>

<bookstore>  
  <book category="children">  
    <title>Harry Potter</title>  
    <author>J K. Rowling</author>  
    <year>2005</year>  
    <price>29.99</price>  
  </book>  
</bookstore>

Un elemento può contenere:

* testo
* attributi
* altri elementi
* un mix di quanto sopra

***Elementi XML vuoti***, si dice che un elemento senza contenuto sia vuoto.

<element></element> = <element /> (tag di chiusura automatica)

Gli elementi vuoti possono avere attributi.

Gli elementi XML devono seguire queste ***regole di denominazione***:

* sono sensibili al maiuscolo / minuscolo
* devono iniziare con una lettera o un trattino basso
* non possono iniziare con le lettere xml (o XML, o Xml, ecc.)
* possono contenere lettere, cifre, trattini, underscore e punti
* non possono contenere spazi

Come questo: <book\_title> e non come questo: <the\_title\_of\_the\_book>.

***Attributi -*** [***https://www.w3schools.com/xml/xml\_attributes.asp***](https://www.w3schools.com/xml/xml_attributes.asp)

Gli attributi sono progettati per contenere dati relativi a un elemento specifico.

I valori degli attributi devono sempre essere citati. È possibile utilizzare virgolette singole o doppie.

<person gender="female"> oppure <person gender='female'>

Alcune cose da considerare quando si usano gli attributi sono:

* non possono contenere più valori (elementi possono)
* non possono contenere strutture ad albero (elementi possono)
* non sono facilmente espandibili (per modifiche future)

A volte i riferimenti ID sono assegnati agli elementi. Questi ID possono essere utilizzati per identificare gli elementi XML più o meno allo stesso modo dell'attributo id in HTML.

<messages>  
  <note id="501">  
    <to>Tove</to>  
    <from>Jani</from>  
    <heading>Reminder</heading>  
    <body>Don't forget me!</body>  
  </note>  
</messages>

Gli attributi id sono per identificare le diverse note. Non fa parte della nota stessa.

I metadati (dati sui dati) dovrebbero essere memorizzati come attributi.

***Namespace -*** [***https://www.w3schools.com/xml/xml\_namespaces.asp***](https://www.w3schools.com/xml/xml_namespaces.asp)

I ***namespace*** XML forniscono un metodo per evitare conflitti tra i nomi degli elementi.

In XML, i nomi degli elementi sono definiti dallo sviluppatore. Ciò si traduce spesso in un conflitto quando si tenta di combinare documenti XML da diverse applicazioni XML.

***Risolvere il conflitto del nome usando un prefisso:***

I conflitti dei nomi possono essere facilmente evitati usando un prefisso del nome: h:table

***L'attributo xmlns:***

Quando si utilizzano prefissi in XML, deve essere definito uno ***spazio*** dei ***nomi*** per il prefisso che può essere definito da un attributo ***xmlns*** nel tag di inizio di un elemento: xmlns: prefisso="URI"***.***

<h:table xmlns:h="http://www.w3.org/TR/html4/">  
  <h:tr>  
    <h:td>Apples</h:td>  
    <h:td>Bananas</h:td>  
  </h:tr>  
</h:table>  
  
<f:table xmlns:f="https://www.w3schools.com/furniture">  
  <f:name>African Coffee Table</f:name>  
  <f:width>80</f:width>  
  <f:length>120</f:length>  
</f:table>

Quando uno spazio dei nomi è definito per un elemento, tutti gli elementi figlio con lo stesso prefisso sono associati allo stesso spazio dei nomi.

Lo scopo dell'utilizzo di un URI è di dare allo spazio dei nomi un nome univoco.

I ***namespace*** possono anche essere dichiarati nell'elemento ***radice*** XML.

***HttpRequest -*** [***https://www.w3schools.com/xml/xml\_http.asp***](https://www.w3schools.com/xml/xml_http.asp)

Tutti i browser moderni dispongono di un oggetto ***XMLHttpRequest*** integrato per richiedere dati da un server.

L'oggetto ***XMLHttpRequest*** è un sogno per gli sviluppatori , perché puoi:

* Aggiorna una pagina Web senza ricaricare la pagina
* Richiedi dati da un server - dopo che la pagina è stata caricata
* Ricevi dati da un server - dopo che la pagina è stata caricata
* Invia dati a un server - in background

[*https://www.w3schools.com/xml/tryit.asp?filename=tryxml\_httprequest*](https://www.w3schools.com/xml/tryit.asp?filename=tryxml_httprequest)

var xhttp = new XMLHttpRequest();  
xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
       // Typical action to be performed when the document is ready:  
       document.getElementById("demo").innerHTML = xhttp.responseText;  
    }  
};  
xhttp.open("GET", "filename", true);  
xhttp.send();

1. La prima riga nell'esempio sopra crea un oggetto ***XMLHttpRequest***:

var xhttp = new XMLHttpRequest();

1. La proprietà ***onreadystatechange*** specifica una funzione da eseguire ogni volta che cambia lo stato dell'oggetto XMLHttpRequest: xhttp.onreadystatechange = function()
2. Quando la proprietà ***readyState*** è 4 e la proprietà **status** è 200, la risposta è pronta: if (this.readyState == 4 && this.status == 200)
3. La proprietà ***responseText*** restituisce la risposta del server come una stringa di testo. La stringa di testo può essere utilizzata per aggiornare una pagina Web:

document.getElementById("demo").innerHTML = xhttp.responseText;

***Parser -*** [***https://www.w3schools.com/xml/xml\_parser.asp***](https://www.w3schools.com/xml/xml_parser.asp)

Il ***DOM XML*** (***Document Object Model***) definisce le proprietà e i metodi per accedere e modificare XML

Tuttavia, prima di poter accedere a un documento XML, è necessario caricarlo in un oggetto DOM XML.

Tutti i browser moderni hanno un parser XML integrato che può convertire il testo in un oggetto DOM XML.

L' [***oggetto XMLHttpRequest***](https://www.w3schools.com/xml/xml_http.asp)ha un parser XML incorporato.

La proprietà ***responseText*** restituisce la risposta come una stringa.

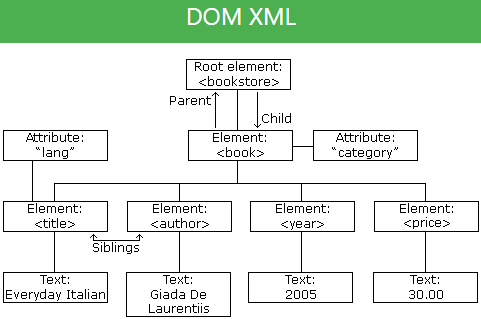
La proprietà ***responseXML*** restituisce la risposta come un oggetto DOM XML.

Se si desidera utilizzare la risposta come oggetto DOM XML, è possibile utilizzare la proprietà ***responseXML***.

<https://www.w3schools.com/xml/tryit.asp?filename=try_dom_xmlhttprequest_responsexml>

***DOM -*** [***https://www.w3schools.com/xml/xml\_dom.asp***](https://www.w3schools.com/xml/xml_dom.asp)

Il DOM XML definisce un modo standard per accedere e manipolare i documenti XML. Presenta un documento XML come *struttura ad albero*.



È possibile accedere a tutti gli elementi XML tramite il DOM XML.

<?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>  
</bookstore>

Questo codice recupera il valore di testo del primo elemento <title> in un documento XML: *txt = xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue*;

Il DOM XML è uno standard su come ottenere, modificare, aggiungere ed eliminare elementi XML.