فصل اول

مقدمه‌ای بر *XML*

۱-۱ *XML* چیست؟

XML مخفف *Extensible Markup Language*، زبانیست که برای محیط وب طراحی شده است. *XML* از زبان نشانه‌دار تعمیم یافتهٔ استاندارد (*Standard Generalized Markup Language*) *SGML* که یک زبان توانمند برای پروژه‌های بزرگ است مشتق شده است. طراحان *XML* آن را با مفاهیم *HTML* کامل نمودند. این زبان جدید تا حدود *20%* از اندازهٔ *SGML* را به خود اختصاص داده، اما به همان اندازه قدرتمند است. اگرچه *SGML* به طور نمونه به وسیلهٔ اشخاصی استفاده شد که به یک زبان با توان صنعتی احتیاج داشتند، اما *XML* برای همهٔ موارد بزرگ و کوچک قابل استفاده است.

XML هیچ تگی از خود ندارد و شما می‌توانید برای اهداف موردنظر خود تگ‌ها را براساس قوانین موجود در *XML* تعریف نمایید. برخلاف *HTML* که تمام عناصر از قبل تعریف شده‌اند و شما فقط می‌توانید از تگ‌های موجود استفاده نمایید که ممکن است برای هدف موردنظر شما کافی نباشد. در واقع *XML* یک ابر زبان است که به شما اجازه می‌دهد تا مواردی را که به آنها احتیاج دارید، تعریف کنید.

این مهم است که بدانید *XML* در مورد محتواست نه در مورد نمایش. تگ‌هایی را که تعریف می‌کنید، روی سازماندهی داده‌های شما متمرکز می‌شوند نه روی نمایش آنها. بدین ترتیب با تعریف تگ‌های مفهومی می‌توانید اطلاعاتی را به سند خود اضافه نمایید، برخلاف *HTML* که نحوهٔ نمایش داده‌ها را تغییر می‌دهد. مثلاً اگر در یک متن مشخصات یک کتاب را بنویسد، با *Bold* کردن نام نویسنده می‌توانید آن را مشخص کنید و کلی باید به دنبال نام نویسنده در صفحه باشید، ولی در *XML* می‌توانید یک تگ برای نام نویسنده ایجاد نمایید.

```
<books>
  <book>
    <title>XSLT Programmers Reference</title>
    <author>Michael Kay</author>
  </book>
</books>
```

با تمام اینها XML جانشینی برای HTML نیست. XML برای توصیف، ذخیره و ارسال داده‌هاست و HTML برای نماش داده‌ها.

۱-۲ برخی ویژگی‌های XML

XML دارای ویژگی‌های متعددی است که مهمترین آنها عبارتند از:

۱. XML مستقل و توسعه‌پذیر است. شما در XML خودتان تگ‌ها را تعریف می‌نمایید و تگ‌ها از پیش تعریف شده نیست.

۲. XML برای مبادلهٔ داده استفاده می‌شود. از آنجایی که XML به صورت متنی می‌باشد و از سخت-افزار و نرم‌افزار جدا است برای انتقال داده بین سیستم‌های ناسازگار می‌تواند استفاده شود.

۳. XML داده را پر استفاده می‌کند. با ساختار ساده‌ای که XML دارد داده می‌تواند در کاربردهای زیادی مورد استفاده قرار گیرد.

۴. XML برای ایجاد زبان جدید می‌تواند مورد استفاده قرار گیرد. XML می‌تواند برای طراحی زبان Wireless Markup Language برای برنامه‌نویسی تلفن همراه مورد استفاده قرار گیرد.

۱-۳ ترکیب XML

ترکیب قوانین XML بسیار ساده و روشن است. یادگیری و استفادهٔ قوانین بسیار آسان بوده و به همین علت ایجاد نرم‌افزاری که بتواند XML را بخواند و به کار ببرد، بسیار آسان است.

در کد زیر یک سند XML نشان داده شده است.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

اولین خط در این سند، اعلان XML است که شامل نسخهٔ XML و کاراکتر کد شده مورد استفاده در سند را توصیف می‌کند. خط بعدی (*<note>*)، عنصر ریشهٔ سند را توصیف می‌کند. در واقع عنصر ریشه نشان‌دهندهٔ موضوع سند بوده و تمام محتویاتی که قرار است در سند توصیف شود، در این عنصر قرار می‌گیرد. همچنین این عنصر شامل یک تگ پایانی (*</note>*) است که در انتهای سند قرار می‌گیرد و انتهای سند را اعلام می‌کند. (کدی که بعد از اعلان انتهایی سند نوشته شود جزء سند محسوب نمی‌شود.) تگ‌های باز (*<to>*)و بسته (*</to>*) برای نشان‌دادن اطلاعات ارسال یادداشت نظیر گیرندهٔ یادداشت، تگ باز (*<from>*)و بسته (*</from>*) برای نشان دادن فرستندهٔ نامه، تگ باز (*<heading>*)و بسته (*</heading>*) برای نشان دادن عنوان نامه و بالاخره تگ باز (*<body>*)و بسته (*</body>*) در بر گیرنده محتویات نامه می‌باشد.

همهٔ عناصر در *XML* باید یک تگ پایانی داشته باشند. اعلان *XML* در خط اول، یک عنصر *XML* حساب نمی‌شود به همین خاطر دارای تگ پایانی نمی‌باشد. نکتهٔ دیگر در مورد *XML* این است که برچسب‌ها در *XML* برخلاف *HTML* به حروف بزرگ و کوچک حساسند و اگر تگ شروع یک عنصر با حروف کوچک یا بزرگ باشد، تگ پایانی آن نیز باید به همان صورت باشد.

```
<Message>this is incorrect</message>   X
<message>this is correct</message>     √
```

در *XML* ساختار تودرتوی عناصر باید به درستی رعایت شود، یعنی اگر یک عنصر درون عنصر دیگر شروع شود، بایستی درون همان عنصر خاتمه یابد.

```
<B><I>I am not bold and italic</B></I>
<B><I>I am bold and italic</I></B>
```

۱-۴ قوانین طراحی اسناد *XML*

طراحی اسناد *XML* دارای یک سری قوانین و مقررات است که با رعایت آنها می‌توان یک سند *XML* درست یا معتبر ساخت. در این قسمت به معرفی این قوانین می‌پردازیم.

۱-۴-۱ همهٔ اسناد *XML* باید یک عنصر ریشه داشته باشند

در تمام اسناد *XML* بعد از اعلان سند، عنصر ریشه قرار می‌گیرد و سایر عناصر باید در درون عنصر ریشه تعریف شوند. عنصر یا عناصری که تعریف آنها باید بعد از تگ انتهایی باشند جزء محتوای سند محسوب نمی‌شوند.

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

۱-۴-۲ مقادیر خاصیت‌ها باید در درون "" قرار گیرند

در تعریف عناصر *XML* می‌توان از خاصیت‌ها برای بیان روشن‌تر تگ‌ها استفاده نمود. در هر تگ دارای خاصیت بعد از عنوان نام، مقدار خاصیت باید بین "" قرار بگیرد، در غیر این‌صورت به‌عنوان مقدار خاصیت در نظر گرفته نمی‌شود.

```
<note date ="12/11/2002"></note>
```

۱-۴-۳ در *XML* فضای خالی حفظ می‌شود

در طراحی *HTML* اگر بین دو کلمه ۱۰ بار از کاراکتر *Space* استفاده کنید، در نتیجه نهایی تنها یک کاراکتر *Space* نمایش داده خواهد شد، ولی در *XML* این قاعده رعایت شده و به همان اندازه که از *Space* استفاده نموده‌اید، آن را در خروجی خواهید دید.

```
Hello     my name is Tove

Hello     my name is Tove                    :XMLفرمت
Hello my name is Tove                        :HTML فرمت
```

۱-۴-۴ در اسناد *XML* از توضیحات استفاده شود

Comment شامل توضیحاتی است که طراح سند برای یادآوری نکته‌ای خاص و یا برای فهم بهتر یک مطلب آن را در زمان طراحی به سند، اضافه می‌نماید.

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

۱-۴-۵ عناصر در *XML* باید قوانین نامگذاری را رعایت نمایند

قوانین نامگذاری عناصر در *XML* عبارتند از:

۱. نام‌ها باید شامل حروف، شماره‌ها و دیگر کارکترها باشند.
۲. نام‌ها نباید با شماره و کارکترهای نقطه‌گذاری شروع شوند.
۳. نام‌ها نباید با حروف *XML* شروع شوند.
۴. نام‌ها نمی‌توانند فضای خالی داشته باشند.

۱-۴-۶ یک خاصیت در *XML* نمی‌تواند دارای چندین مقدار باشد

در *XML* به هر خاصیتی که تعریف می‌کنید، فقط یک مقدار می‌توانید تخصیص دهید. مثلاً در مثالی که در بخش ۲-۴-۱ برای مقداردهی خاصیت‌ها گفته شد، نمی‌توان خاصیت موردنظر را به صورت زیر مقداردهی کرد.

```
<note date ="12/11/2002" , "12/10/2002"></note>
```

۱-۵ *Well-Formed XML* یا *XML* خوش ترکیب

علاوه بر قوانینی که در طراحی یک فایل *XML* (قوانین معرفی شده در قسمت ۴-۱) باید آنها را رعایت نمایید، قوانین دیگری نیز وجود دارند که برای ساخت یک *XML* خوش ترکیب یا *Well-Formed* باید آنها را رعایت نمایید. برخی از مفاهیم و قوانینی که در ساخت یک متن *XML* خوش ترکیب باید رعایت شوند عبارتند از:

۱. سند باید حاوی یک یا چند *Element* یا عنصر باشد.
۲. سند باید شامل فقط یک عنصر ریشه باشد.
۳. عنصر پایانی یا آخرین عنصر باید همنام با عنصر شروع باشد.
۴. در یک عنصر، یک خاصیت نباید بیشتر از یکبار تکرار شود.
۵. مقادیر خاصیت‌ها نباید شامل کاراکتر < باشند.
۶. عناصر با تگ‌های شروع و پایان شناخته می‌شوند که باید به درستی در درون هم قرار گرفته باشند.

۱-۶ تکنولوژی‌های *XML*

بعد از معرفی *XML* و قوانین طراحی آن به معرفی و نحوهٔ کار با تکنولوژی‌هایی که همراه با *XML* مورد استفاده قرار می‌گیرند، می‌پردازیم. هر کدام از این تکنولوژی‌ها کاربرد خاص خود را دارند:

DTD: برای تعریف الگوی فایل *XML* مورد استفاده یا به عبارتی برای اعتبارسنجی داده‌هایی که کاربر به یک سند *XML* می‌فرستد، مورد استفاده قرار می‌گیرد.

Schema: کاربرد *Schema* نیز دقیقاً مانند *DTD* می‌باشد و از آن نیز برای اعتبار سنجی و تعریف ساختار یک سند *XML* استفاده می‌شود تا کاربرانی که از فایل استفاده می‌کنند، بدانند چگونه داده‌های خود را تعریف نمایند.

نکته: *یک سند معتبر XML سندی است که مطابق با قوانین تعریف شده در DTD و Schema باشد.*

XSLT: با استفاده از *XSLT* شما می‌توانید فایل *XML* را به فرمت‌های موردنظرتان تبدیل نمایید و خروجی‌ای را تولید کنید که بتوان آن را در مرورگر وب نمایش داد. *W3C*، *XSLT* را زبانی برای تبدیل اسناد *XML* به اسناد *XML* دیگر توصیف می‌نماید، اما کاربرد *XSLT* بیشتر از آن است. رابطهٔ *XSLT* و *XML* مانند رابطهٔ *SQL* و *DataBase* می‌باشد. همان‌طوری‌که *SQL* می‌تواند *Query* بگیرد و داده‌ها را تغییر دهد، *XSLT* نیز می‌تواند از بخشی از سند *XML*، *Query* بگیرد و محتویات جدید تولید نماید.

XPath: *XML* از لحاظ تکنیکی دارای این محدودیت است که برای ایجاد *Query* از داده‌های *XML*، نمی‌توانید به تنهایی از خود *XML* استفاده نمایید. با *XPath* این محدودیت رفع شده است. *XPath* یک زبان پرس‌وجوی هدایت‌گر مخصوص می‌باشد که برای قرار دادن داده در یک سند *XML* به کار گرفته شده است. شما می‌توانید از *XPath* برای ایجاد *Query* از اسناد *XML* استفاده نمایید، همان‌طوری که از *SQL* برای *Query* گرفتن از *DataBase* استفاده می‌نمایید. *XPath* می‌تواند از یک قسمت از سند یا عناصر، خاصیت‌ها و یا مقادیر موجود در سند *XML*، *Query* بگیرد و اطلاعات خاصی را که در پرس و جوی طراحی شده صدق می‌کند برگرداند.

DOM: *DOM* را در *Application*های خود تعریف می‌کنید تا بتوان به سند *XML*ی که طراحی شده دسترسی داشته و آن را تغییر دهید.

۱-۶-۱ *DTD*

در صورتی که هنگام طراحی یک فایل *HTML* اشتباهی وجود داشته باشد، مرورگر وب از نشان دادن صفحهٔ نمایش خودداری می‌کند، اما اگر در طراحی *XML* اشتباهی رخ دهد (مثلاً کاربر تگ پایانی یک برچسب را فراموش نماید) چه کسی متوجه خواهد شد؟

همچنین از آنجایی که کسی از ساختار موردنظر شما در زمان طراحی فایل *XML* اطلاعی ندارد، نمی‌تواند شما را از اشتباه رخ داده با خبر سازد.

برای حل این مشکلات در *XML* ابزارهای متعددی پیش‌بینی شده است. مثلاً می‌توان از فایل‌های *Definition Document Type* یا به اختصار *DTD* استفاده کرد. به کمک این نوع فایل می‌توانید بدون توجه به اصل داده‌ها، فقط شکل و ساختار سازمانی داده‌های خود را در آن تعریف نمایید. در واقع در *DTD* کاربر تعیین می‌کند که در فایل *XML* چه برچسب‌ها و چه داده‌هایی به کار گرفته شده‌اند و چه ویژگی‌هایی دارند.

از *DTD* می‌توان برای تعیین اعتبار فایل *XML* قبل از نمایش آن استفاده نمود. تعیین صحت ساختاری یک فایل *XML* و یا تطابق ساختار آن با آنچه در فایل *DTD* متناظر بیان می‌شود، *Validity* یا اعتبار سنجی نام دارد.

DTD یا *Document Type Definition* در واقع ساختار و نوع داده‌ای عناصر مختلف یک فایل *XML* را مشخص می‌کند. *DTD* مثل بخش تعریف متغیر در یک برنامهٔ پاسکال یا *C* می‌باشد، چرا که در این بخش از برنامه نوع متغیر مورد استفاده و گاهی مثل تعریف آرایه‌ها ساختار آنها نیز مشخص شده و در متن برنامه از این متغیرهای از پیش تعریف شده استفاده می‌شود. رابطهٔ *DTD* و فایل *XML* نیز چنین می‌باشد، یعنی ابتدا در بخش *DTD* نوع عناصر، نوع مقادیر آنها و ساختار آنها مشخص شده و در قسمت داده‌ای فایل *XML* که داده‌ها وارد می‌شوند، این عناصر تعریف شده مورد استفاده قرار می‌گیرند.

> ***نکته:*** *بدیهی است که در یک ساختار استاندارد، استفاده از عناصری که در DTD تعریف نشده‌اند، مجاز نیست.*

DTD می‌تواند هم به صورت درونی *Inline* و هم به صورت یک فایل خارجی (*ExternalFile*) به فایل *XML* ما مرتبط شود.

چند دلیل ساده برای استفاده از *DTD*:

سه دلیل مهم استفاده از *DTD* عبارتند از:

۱. هر فایل *XML* می‌تواند خودش به راحتی ساختار و نوع اطلاعات درونش را تعریف کند.
۲. وقتی ساختار یک فایل *XML* توسط *DTD* مشخص شود، نرم‌افزارهای مختلفی که از نوع فایل *XML* استفاده می‌کنند می‌توانند براساس این استاندارد تعریف شده در *DTD*، اطلاعات را با این ساختار مشخص ما بین یکدیگر ردوبدل کنند(مثل وب سرویس‌ها).
۳. نرم‌افزار شما که این فایل *XML* را دریافت می‌کند، به راحتی با چک‌کردن *DTD* و عناصر فایل *XML* می‌تواند تشخیص بدهد که آیا اطلاعات موجود در فایل دریافتی صحیح می‌باشد یا خیر؟

> **نکته:** *برای راحت‌تر خوانده‌شدن فایل XML و جدا کردن داده‌ها از ساختارشان می‌توانید یک فایل DTD تعریف کنید و قسمت‌های تعریف عناصر و خاصیت‌های آنها را در فایل جداگانه قرار داده و آن را با فایل XML پیوند دهید.*

اکنون یک مثال فایل *XML* به همراه *DTD* آن را برای شما تشریح می‌کنیم.

```
<?xml version="1.0" ?>
<!DOCTYPE note [
<!ELEMENT  note (to,from,heading,body)>
<!ELEMENT to(#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
  <to>Tove</to>
  <from>Jani</from>
```

```
  <heading>Reminder</heading>
  <body>Don't forget me this weekend</body>
</note>
```

در مثال بالا قسمتی که بین *[و]* قرار گرفته، *DTD* فایل *XML* می‌باشد. اکنون قسمت‌های مختلف *DTD* نمایش داده شده در مثال را تشریح می‌کنیم:

همان‌طور که مشخص است، نوع *DTD* به کار رفته در مثال به صورت *Inline* می‌باشد. با در نظر گرفتن این نکته ساختار تعریف *DTD* به صورت *Inline* را به صورت زیر تعریف می‌کنیم:

```
<!DOCTYPE root-element [element-declaration] >
```

کلمهٔ *!DOCTYPE* مشخص می‌کند که اطلاعات ذخیره شده در فایل *XML* در مورد چیست و با توجه به اینکه همیشه *root-element* مشخص‌کننده این مطلب است، در مثال بالا در قسمت *!DOCTYPE node* مشخص شده که اطلاعات درون فایل *XML* در مورد *note* است.

در قسمت *element-declaration* ساختار و نوع داده‌ای عناصر فایل *XML* را تعریف می‌کنیم. در این مثال *<!ELEMENT note (to, from, heading, body)>* مشخص شده است که عنصر *note* خود دارای ۴ عنصر فرزند (*child element*) با نام‌های *to, from, heading,body* می‌باشد. در ادامه کد *<!ELEMENT to (#PCDATA)>* مشخص شده است که عنصر *to* دارای نوع *PCDATA* می‌باشد. در این کدها کلمات کلیدی *!ELEMENT* برای تعریف یک عنصر و *#PCDATA* برای مشخص‌کردن یک نوع داده‌ای است (در ادامه توضیح داده می‌شود).

اگر بخواهیم *DTD* همین مثال را به صورت یک فایل خارجی نشان دهیم، فایل *XML* به شکل زیر خواهد شد:

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

در این مثال *<!DOCTYPE note SYSTEM "note.dtd">*، قسمتی است که تعیین‌کنندهٔ *DTD* سند *XML* می‌باشد، در یک فایل خارجی به نام *note.dtd* قرار دارد. با این تفاسیر نحوهٔ اتصال یک فایل *XML* به یک فایل *DTD* خارجی به صورت زیر است:

```
<!DOCTYPE root-element SYSTEM " DTD مسیر و نام فایل ">
```

اگر فایل *DTD* در محل ذخیره‌سازی فایل *XML* باشد، فقط ذکر نام آن کافی است در غیر این صورت باید مسیر آن را نیز مشخص کنیم. اما در اینجا سؤالی که مطرح است آن است که محتوی فایل *note.dtd* در مثال این چه می‌باشد؟

پاسخ این سؤال محتوی فایل *note.dtd* (قسمت *element-declaration*) در قسمت قبل است.

در سطر اول این کد، عنصر ریشه به همراه عناصر زیر شاخهٔ آن که در قسمت () قرار گرفته‌اند، و در ۴ سطر بعدی هر کدام از عناصر زیر شاخه نشان داده شده است.

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

در مثال‌های فوق کلمهٔ *#PCDATA* مشخص‌کنندهٔ رشتهٔ متنی است که در آن کاراکترهای رزرو شده وجود ندارد و در صورت وجود به معادلشان تغییر داده شده‌اند. این کاراکترها که لیست آنها در جدول ۱-۱ آمده است، برای تعریف و جداسازی عناصر و ساختارها از داده‌های فایل *XML* به کار می‌روند (به آنها *Entity* گفته می‌شود) و در صورتی که بخواهیم از این کارکترها به عنوان داده استفاده کنیم، باید معادل آنها را به کار ببریم.

جدول ۱-۱

کارکتر	معادل
<	*<*
>	*>*
&	*&*
"	*"*
'	*'*

نوع دیگر که کمتر به کار می‌رود *#CDATA* می‌باشد. این نوع نیز مانند *#PCDATA* دادهٔ رشته‌ای متنی است با این تفاوت که درون آن می‌تواند کارکترهای غیرمجاز وجود داشته باشد، به همین دلیل وقتی محتوای یک عنصر را *#CDATA* تعریف می‌کنیم، آن عنصر هیچ گونه عنصر فرزندی نمی‌تواند داشته باشد زیرا توسط نرم‌افزارهای مفسر *XML* نادیده گرفته خواهند شد.

اکنون می‌خواهیم بحث تعریف عناصر را به صورت کامل بررسی کنیم. به همین منظور ابتدا نحوهٔ تعریف یک عنصر در *DTD* را بررسی کرده و سپس انواع تعریف آن را با مثال مشخص می‌کنیم.

چنانچه تاکنون مشاهده نموده‌اید، نحوهٔ تعریف یک عنصر در *DTD* به صورت زیر بوده است:

<!ELEMENT نام عنصر *(Element Type or Element Content)>*

"نام عنصر" که کاملاً مشخص می‌باشد، همان نامی است که در فایل *XML* در قسمت داده‌ای به کار می‌رود. انواع عناصری که در زبان *XML* با آن سروکار دارید، عبارتند از:

- **عنصر تهی یا بدون محتوا**

در صورتی‌که بخواهیم در *DTD* عنصری تعریف کنیم که هیچ مقدار یا محتوایی نداشته باشد از قالب زیر استفاده می‌کنیم:

<!ELEMENT نام عنصر EMPTY>

مثال:

```
<!ELEMENT br EMPTY>
```

در این مثال عنصری به نام *br* تعریف شده که هیچ محتوایی ندارد (این عنصر همان تگ *br* در *HTML* می‌باشد که باعث شکسته شدن خط متن به خط بعدی می‌شود).

نکته: *عناصری که هیچ محتوایی ندارند، در قسمت داده‌ای فایل XML برخلاف سایر انواع عناصر که دارای یک تگ ابتدایی و یک تگ انتهایی هستند فقط دارای تگ ابتدایی می‌باشد.*

همانند عنصر *br* که در مثال قبل تعریف کردیم (یک عنصر تهی یا عنصر بدون محتوا)، در یک فایل *XML* (قسمت داده‌ای فایل) به شکل زیر نمایش داده می‌شود:

```
<br/>
```

- **عنصر *#PCDATA*:**

در صورتی که بخواهیم در *DTD* عنصری تعریف کنیم که محتوای آن از نوع *#PCDATA* باشد، از قالب زیر استفاده می‌کنیم:

```
<!ELEMENT نام عنصر (#PCDATA)>
```

مثال:

```
<!ELEMENT from (#PCDATA)>
```

- **عنصر آزاد**

در صورتی که بخواهیم در *DTD* عنصری تعریف کنیم که هر نوع محتوایی را بپذیرد، از قالب زیر استفاده می‌کنیم:

```
<!ELEMENT نام عنصر ANY>
```

مثال:

```
<!ELEMENT note ANY>
```

- عنصر والد شامل عنصر فرزند یا (*Child Element*)

چنانچه بخواهیم در *DTD* عناصری را تعریف کنیم که خود شامل زیر شاخه، عنصر فرزند و یا *Child Element* باشد، قالب تعریف آن به صورت زیر خواهد بود:

```
<!ELEMENT نام عنصر (1 نام عنصر فرزند , 2 نام عنصر فرزند , ...) >
```

مثال:

```
<!ELEMENT note (to,from,heading,body)>
```

در این مثال عنصر *note* دارای چند عنصر فرزند است که بین (و) قرار گرفته‌اند.

مثال:

```
<!ELEMENT note (message)>
```

در این مثال عنصر *note* (والد) دارای یک عنصر فرزند می‌باشد. بعد از عنوان کلمهٔ *ELEMENT*، نام عنصر فرزند(*note*) و مقدار آن (*message*) نوشته می‌شود.

- **عنصر والد شامل یک یا چند عنصر فرزند**

چنانچه بخواهیم عنصری تعریف کنیم که عنصر فرزند آن یک بار یا بیشتر درون عنصر والد تکرار شود، در قسمت سمت راست نام عنصر فرزند علامت "+" را قرار می‌دهیم.

```
<!ELEMENT نام عنصر پدر (نام عنصر فرزند +)>
```

مثال:

```
<!ELEMENT book (chapter+)>
```

در این مثال چون یک کتاب می‌تواند چندین بخش داشته باشد پس *chapter* که عنصر فرزند عنصر *book* است همراه با کاراکتر + استفاده شده است. به این معنا که این کتاب می‌تواند یک یا چندین بخش داشته باشد.

- **عنصر والد شامل هیچ، یک یا چند عنصر فرزند**

در صورتی‌که بخواهیم عنصر والدی که عنصر فرزند آن می‌تواند وجود نداشته باشد، یا چندین بار درون عنصر والد تکرار شود تعریف کنیم، در قسمت سمت راست نام عنصر فرزند، علامت "*" را قرار می‌دهیم.

```
<!ELEMENT نام عنصر پدر (نام عنصر فرزند*)>
```

مثال:

```
<!ELEMENT Father (boy*)
```

در این مثال چون عنصر والد می‌تواند هیچ یا یک عنصر فرزند داشته باشد، پس عنصر *boy* می‌تواند درون عنصر *father* اصلاً تکرار نشود یا چندین بار تکرار شود.

- **عنصر والدی که از بین چند عنصر فرزند می‌تواند فقط یکی را داشته باشد**

چنانچه بخواهیم عنصری تعریف کنیم که محتوای آن به تعدادی عنصر فرزند محدود شده باشد (شامل زیر شاخه‌هایی باشد) و در هر تعریف به طور انتخابی یکی از این عناصر فرزند را بتواند استفاده نماید، قالب تعریف آن به صورت زیر خواهد بود:

```
<!ELEMENT نام عنصر والد (عنصر فرزند ۱ | عنصر فرزند ۲) >
```

مثال:

```
<!ELEMENT note (to,from,header,(message|body))>
```

در این مثال عنصر *note* درای عناصر فرزند *to,from,header* می‌باشد ولی در آن واحد از بین عناصر *message* و *body* فقط می‌تواند یکی را داشته باشد.

- **عنصری با محتوای ترکیبی (*Mixed Content*)**

چنانچه بخواهیم عنصری تعریف کنیم که محتوای آن ترکیبی از نوع داده‌ای و عناصر فرزند باشد، قالب تعریف آن به صورت زیر خواهد بود:

```
<!ELEMETNT نام عنصر والد (نوع داده‌ای ۱, نام عنصر فرزند ۲, نام عنصر فرزند ۱, . . .)
```

مثال:

```
<!ELEMENT chapter (#PCDATA, caption+) >
```

در این مثال عنصر *chapter* خود دارای محتوا از نوع *#PCDATA* می‌باشد و در ضمن دارای عنصر فرزند *caption* است که می‌تواند یک یا چند بار تکرار شود:

```
<chapter>this is an Excellent chapter
<caption>there is a problem</caption>
<caption>what is Excellent</caption>
<caption>there is a problem</caption>                    </chapter>
```