Krótka historia języka HTML

Być może spotkałeś się już z określeniem, że HTML to „martwy" język. Jak mawiał Douglas Adams, „Nie panikuj!". Dotychczas ukazało się już 5 specyfikacji HTML-a: 1.0, 2.0, 3.2, 4.0 i 4.01 i nigdy nie było tak, że nowa wersja wyeliminowała z rynku poprzednie. Gdyby za każdym razem trzeba było przeprojektowywać istniejące strony WWW, Internet szybko uległby ogromnemu rozdrobnieniu. Dlatego przeglądarki internetowe obsługują bezproblemowo wszystkie odmiany języka HTML. To, że nigdy nie ukaże się już wersja 5.0 nie oznacza wcale, że język ten jest martwy. Po prostu ewoluuje on w taki sposób, aby współgrał z nowymi narzędziami i technologiami.

Konsorcjum W3C wypuściło ostatnią specyfikację języka HTML w Wigilię Bożego Narodzenia w 1999 roku i było to jej ostatnie słowo w kwestii języka, który funkcjonował w Internecie już ponad 10 lat. HTML nie był ani idealnym, ani eleganckim rozwiązaniem, ale dobrze realizował powierzone przed nim zadania. Bez niego sieć WWW w ogóle by nie istniała i nie urosłaby do takich rozmiarów. W specyfikacji 4.01 twórcy języka uznali, że najlepiej będzie oddzielić wygląd dokumentu od jego treści, wprowadzając Kaskadowe Arkusze Stylów (CSS). Pozwalały one obchodzić ograniczenia języka HTML i zwiększały możliwości projektantów witryn.

HTML jest tak naprawdę podzbiorem języka SGML. Ten ostatni jest zbiorem reguł mówiących czym powinny się charakteryzować języki znaczników stosowane w informatyce. HTML miał być z założenia szybki i łatwy do opanowania, dlatego jego zgodność z językiem SGML jest dość przypadkowa. Trzeba też powiedzieć, że twórcy przeglądarek internetowych wprowadzali często własne, niestandardowe znaczniki. Nie powinien więc dziwić fakt, że język HTML często zbaczał z nadanej mu ścieżki. Z każdą kolejną wersją specyfikacji konsorcjum W3C starało się przywołać HTML-a do porządku i powrócić do zgodności z językiem SGML. Niestety starania te zakończyły się jedynie częściowym sukcesem. Po prostu SGML jest zbyt rozbudowanym standardem i nie ma takiej możliwości, aby język HTML był z nim w pełni zgodny, a przy tym pozostawał językiem, którego typowy użytkownik mógłby się łatwo nauczyć. Wymyślono więc rozwiązanie leżące gdzieś między sztywnymi regułami SGML-a, a pożądaną elastycznością i prostotą.

XHTML na ratunek

Język XML stanowi pierwszy krok na drodze ku znalezieniu idealnego rozwiązania między SGML-em a HTML-em. Podobnie jak SGML, język XML (który się z niego wywodzi) jest metajęzykiem, ale można się go łatwo nauczyć. W przeciwieństwie do

HTML-a jest on w pełni zgodny z językiem SGML. Nie jest to typowy język znaczników, lecz zestaw reguł, w oparciu o które można przekształcić dowolny zbiór logicznie uporządkowanych informacji w dokument tekstowy oparty na znacznikach.

W języku XML najpierw tworzy się element opisujący podstawowy typ danych wykorzystywany przez użytkownika. Aby zrozumieć, o co chodzi, wyobraź sobie bazę danych zawierającą informacje o książkach znajdujących się w magazynie księgarni. Wszystkie dane o zapasach są logicznie uporządkowane i zawierają następujące informacje:

- Tematyka książki
- Tytuł książki
- Nazwisko autora
- Wydawca
- ISBN
- Liczba stron
- Cena detaliczna

Wyobraź sobie więc, że taka baza danych zawiera siedem pól. Oto jak można je przedstawić w języku XML:

```
<ksiazka tematyka="komputery">
   <ksiazka.tytul>HTML w 10 prostych krokach</ksiazka.tytul>
   <autor>
      <imie>Robert</imie>
      <nazwisko>Fuller</nazwisko>
   </autor>
   <wyd>Helion</wyd>
   <isbn>0764541234</isbn>
   <stron>600</stron>
   <cena.detaliczna>50.00</cena.detaliczna>
</ksiazka>
```

Ten przykładowy dokument w języku XML zawiera element główny, zbudowany z elementów składowych. Ma on swoje atrybuty, co sprawia, że dokument przypomina plik HTML. Jednak w przeciwieństwie do znaczników języka HTML, znaczniki XML-a nie określają sposobu, w jaki poszczególne informacje mają być wyświetlane na ekranie. Co więcej, sposób ich interpretacji zależy tylko i wyłącznie od aplikacji, w której są one odczytywane. Język XML nie gwarantuje nawet tego, że zostaną one w ogóle wyświetlone na ekranie. To, w jaki sposób zostaną wykorzystane opisane za jego pomocą informacje zależy tylko od użytkowników, którzy mają do nich dostęp.

Jednak w języku XML nie chodzi jedynie o to, aby wymyślić własny zestaw znaczników. Zanim będzie można z nich korzystać, trzeba jeszcze zdefiniować wszystkie elementy, jakie mają prawo pojawić się w dokumencie i ich atrybuty. Do tego służy właśnie definicja typu dokumentu (DTD), zawierająca informacje o wszystkich dozwolonych elementach i ich częściach składowych. Przykładowa definicja DTD wygląda następująco:

```
<!ELEMENT ksiazka ( ksiazka.tytul, autor+, wyd, isbn, stron, cena.detaliczna )>
<!ELEMENT ksiazka.tytul (#PCDATA) >
<!ELEMENT autor ( imie, nazwisko ) >
```

```
<!ELEMENT imie ( #PCDATA ) >
<!ELEMENT nazwisko ( #PCDATA ) >
<!ELEMENT wyd ( #PCDATA ) >
<!ELEMENT isbn ( #PCDATA ) >
<!ELEMENT stron ( #PCDATA ) >
<!ELEMENT cena.detaliczna ( #PCDATA ) >
<!ATTLIST ksiazka tematyka (komputery|beletrystyka|literaturafaktu) #REQUIRED >
```

Za pomocą deklaracji <!ELEMENT> definiuje się poszczególne znaczniki. Wartości umieszczone w nawiasach informują o tym, jakiego rodzaju informacje są umieszczane między znacznikiem otwierającym i zamykającym. Na przykład niektóre elementy, takie jak autor, zawierają jeszcze elementy składowe, a inne, takie jak isbn, przechowują jedynie dane.

Jak widać na powyższym przykładzie, element <ksiazka> ma atrybut tematyka. Atrybuty definiuje się za pomocą deklaracji <!ATTLIST>, w których podaje się nazwę elementu, nazwę atrybutu i ujętą w nawiasy listę wartości, które dany atrybut może przyjmować. W przytoczonej przeze mnie deklaracji DTD atrybut tematyka jest zdefiniowany jako obligatoryjny (ang. required). Oznacza to, że jeśli ktoś wprowadzi do dokumentu element niezawierający atrybutu tematyka, dokument nie zostanie uznany za prawidłowy.

Definicje DTD można dołączać do dokumentów XML lub osadzać je bezpośrednio w nich. Oto jak może wyglądać taki przykładowy dokument w języku XML:

```
<!DOCTYPE ksiazka [
<!ELEMENT ksiazka ( ksiazka.tytul, autor+, wyd, isbn, stron, cena.detaliczna )>
<!ELEMENT ksiazka.tytul (#PCDATA) >
<!ELEMENT autor ( imie, nazwisko ) >
<!ELEMENT imie ( #PCDATA ) >
<!ELEMENT nazwisko ( #PCDATA ) >
<!ELEMENT wyd ( #PCDATA ) >
<!ELEMENT isbn ( #PCDATA ) >
<!ELEMENT stron ( #PCDATA ) >
<!ELEMENT cena.detaliczna ( #PCDATA ) >
<!ATTLIST ksiazka tematyka (komputery|beletrystyka|literaturafaktu)
]
>

<!—Deklaracja DOCTYPE może albo zawierać listę deklaracji elementów, albo adres URL
prowadzący do dokumentu zawierającego DTD, na przykład "ksiazki.dtd" -->

<sklep.magazyn>
<tytul>Magazyn Ksiegarni SuperBooks</tytul>
<ksiazka tematyka="komputery">
<ksiazka.tytul>HTML w 10 prostych krokach</ksiazka.tytul>
<autor>
<imie>Robert</imie>
<nazwisko>Fuller</nazwisko>
</autor>
<wyd>Helion</wyd>
<isbn>1234567</isbn>
<stron>600</stron>
<cena.detaliczna>50.00</cena.detaliczna>
</ksiazka>
</sklep.magazyn>
```

Pierwszy wiersz, <?xml version="1.0"?>, określa typ dokumentu (w tym przypadku jest to dokument w języku XML) i wersję języka, z którą jest on zgodny. Nie jest to żaden element, ani znacznik, lecz instrukcja sterująca przeznaczona dla aplikacji, która będzie przetwarzała dane. Tuż za nią znajduje się definicja DTD, która w tym przypadku jest osadzona bezpośrednio w pliku XML (Znacznik <!DOCTYPE> mógłby też zawierać adres URL do zewnętrznego pliku DTD, o rozszerzeniu .dtd). Następnie zadeklarowany jest element główny — sklep.magazyn. Wewnątrz niego znajduje się element <tytul>, po którym występuje tyle elementów, ile jest książek na magazynie.

To, co tu przedstawiłem, to jedynie mały wycinek tego, co można zrobić za pomocą języka XML. Jest to język zgodny z CSS, a więc dla jego elementów można utworzyć kaskadowe arkusze stylów i wyświetlać je w przeglądarce. XML ma też własny język do tworzenia arkuszy stylów, XSL, który jest takim przeskokiem w stosunku do CSS, jakim XML jest w stosunku do HTML-a.

Mieszanina języków XML i HTML, czyli XHTML

Konsorcjum W3C uznało w pewnym momencie, że istnieje realne zagrożenie, że język HTML stanie się niekontrolowaną mieszaniną przypadkowych znaczników pochodzących z różnych źródeł, nad którą będzie bardzo trudno zapanować. Dlatego trzeba było podjąć kroki zmierzające do uporządkowania tego „bałaganu" i przywołania języka HTML do porządku poprzez przywrócenie mu zgodności z językiem SGML. Pojawiło się jedynak pytanie, czy nie doprowadzi to do takiego skomplikowania tego języka, że przestanie się on rozwijać, bo będzie zbyt trudny, aby ludzie chcieli z niego korzystać. Okazuje się, że nie. Wystarczyło bowiem zdefiniować HTML na nowo — tym razem w języku XML.

W styczniu 2000 roku konsorcjum W3C wypuściło rekomendację XHTML 1.0. Od tej chwili XHTML stał się oficjalnym językiem znaczników sieci WWW. W3C stworzyło dokument DTD definiujący istniejące znaczniki i atrybuty języka HTML, z tym że pominięto te spośród nich, których używanie nie jest już zalecane ze względu na dostępność arkuszy CSS. Zaadaptowano też składnię języka XML i udostępniono możliwość tworzenia nowych znaczników. Dzięki temu język XHTML może się rozwijać. Jeżeli potrzebujesz nowego znacznika, możesz po prostu napisać definicję DTD, umieścić w swoim dokumencie XHTML odwołanie do niej, utworzyć w pliku CSS regułę stylu dla nowego znaczniku i zacząć go używać.

Ale co z językiem HTML?

Pomimo wszystkich tych stwierdzeń, że HTML jest „martwy", na razie nie wydaje się, aby wybierał się on do grobu. Jeszcze przez bardzo długi czas będzie on obsługiwany przez przeglądarki internetowe, która zamiast myśleć o jego porzuceniu powinny raczej skupić się na jego lepszej obsłudze, ponieważ na dzień dzisiejszy żadna przeglądarka

nie jest w pełni zgodna z obowiązującym standardem HTML 4.01. Oczywiście standard XHTML 1.0 też nie jest jeszcze powszechnie obsługiwany, ale wszyscy liczący się producenci przeglądarek dbają o to, aby każda kolejna wersja ich programu była bardziej zgodna z tym standardem od poprzednich.

Różnice w składni języków XHTML i HTML

Języki XHTML i HTML mają bardzo podobną składnię. Niektóre przeglądarki pozwalają na dość swobodne podchodzenie do jej reguł, ale my zawsze woleliśmy trzymać się oficjalnego standardu, ponieważ jest to dobre zabezpieczenie na przyszłość. Niżej wymieniamy główne różnice w składni obu opisywanych języków:

- **Wielkość znaków:** We wszystkich przykładowych kodach źródłowych zamieszczonych w książce nazwy znaczników zapisane są małymi literami. Zachęcamy Cię, abyś Ty też tworzył dokumenty HTML w taki sposób, ponieważ język XHTML rozróżnia wielkie i małe litery i wymaga, aby nazwy wszystkich znaczników, atrybutów, a także reguł CSS były zapisywane małymi literami. Oznacza to, że w języku XHTML <p> i <P> to dwa różne znaczniki, a width i WIDTH to dwa różne atrybuty.
- **Znaczniki zamykające:** W języku XHTML należy zawsze stosować znaczniki zamykające. W przeciwieństwie do języka HTML, który pozwala na ich pomijanie w sytuacji, gdy przeglądarka może się łatwo domyślić, gdzie powinny się one znaleźć, język XHTML nie dopuszcza takiej możliwości. Dokumenty bez znaczników zamykających nie są prawidłowo przetwarzane.
- **Puste znaczniki:** W języku XHTML wymóg stosowania znaczników zamykających jest posunięty tak daleko, że nawet tak zwane puste znaczniki, które w HTML-u nie mają znaczników zamykających, muszą być zamykane poprzez umieszczanie tuż przed nawiasem zamykającym znaku ukośnika. Na przykład:

```
<base attribute="wartość"/>
<basefont attribute="wartość"/>
<br attribute="wartość"/>
<frame attribute="wartość"/>
<hr attribute="wartość"/>
<img attribute="wartość"/>
<input attribute="wartość"/>
<link attribute="wartość"/>
```

  Oczywiście gdy przeglądarka nieobsługująca języka XHTML natrafi na tego rodzaju znaczniki, może pojawić się mały problem. Nie będzie ona wiedziała, w jaki sposób należy je obsłużyć. Można jednak sobie z tym poradzić, umieszczając między właściwym znacznikiem, a sekwencją zamykającą /> znak spacji. Na przykład:

```
<br />
<hr />
<img src="kot.gif" alt="Lew" />
```

- **Wartości atrybutów w cudzysłowach:** W języku XHTML wartości wszystkich atrybutów muszą być umieszczane w cudzysłowach. W języku HTML w przypadku niektórych wartości numerycznych nie było takiej potrzeby. Na przykład poniższy fragment kodu z punktu widzenia języka HTML jest jak najbardziej poprawny:

```
<img src="images/kot.gif" width=50 height=100 border=0>
```

  Zalecamy umieszczanie w cudzysłowach wszystkich wartości nadawanych atrybutom, ponieważ tak zapisane dokumenty są spójne, a poza tym pozwala to zaoszczędzić mnóstwo pracy w przyszłości, kiedy pojawi się potrzeba przekształcenia dokumentów z języka HTML na XHTML.

- **Jednowyrazowe wartości atrybutów:** W języku HTML występuje wiele atrybutów, którym nie trzeba przypisywać żadnych wartości. Ich nazwy same w sobie stanowią informację dla przeglądarki, że powinna się ona zachować ten czy inny sposób. Na przykład atrybut noshade znacznika <hr> lub atrybut selected attribute znacznika <input>, dzięki któremu wiadomo, która opcja z kilku dostępnych ma być wstępnie zaznaczona. W języku XHTML wszystkie atrybuty muszą być definiowane w następujący sposób: atrybut="wartość". Oznacza to, że wspomniany atrybut noshade musi zostać zapisany jako <hr noshade="noshade">.

- **Znaki specjalne:** Znak ampersand (&) i znaczniki komentarzy znane z języka HTML (<!-- -->) w języku XHTML są źródłem problemów. Ampersandy można umieszczać wewnątrz innych elementów, takich jak akapity, czy komórki tabeli, ale nie można stosować ich „luzem" w charakterze wartości przypisywanych do atrybutów. Na przykład wyrażenie alt="War & Peace" byłoby już problematyczne. Jednak istnieje obejście tego problemu. Wystarczy stosować opisowy odpowiednik znaku &, to znaczy wyrażenie (&).

  Zaś jeśli chodzi o znaczniki komentarzy, to stanowią one problem w przypadku osadzanych arkuszy stylów i skryptów, których kod jest ukrywany w dokumentach HTML właśnie za ich pomocą. Dzięki temu jest on niewidoczny dla starszych przeglądarek, które go nie obsługują. Na przykład:

```
<style type="text/css">
<! --

// Tu znajduje się kod CSS

-->
</style>
```

  lub

```
<script language="JavaScript">
<!--

// Tu znajduje się kod w języku JavaScript

//-->
</script>
Najprostszym rozwiązaniem jest zastosowanie znacznika <link>. Na przykład:
```

```
<link rel="stylesheet" type="text/css"
href="http://www.witryna.com/moje_style.css">
```

lub

```
<script language="JavaScript" src="http://www.witryna.com/rollover.js"
type="text/javascript">
</script>
```

- **Atrybut id:** Gdy pracujesz z „kotwicami", zamiast atrybutu name możesz stosować atrybut id. Dawniej w języku HTML atrybut name był też niezbędny przy definiowaniu kontrolek formularzy, na przykład w znaczniku <input>. W specyfikacji HTML 4.0 pojawił się jednak atrybut id, który działa dokładnie tak samo jak name. W specyfikacji XHTML 1.0 korzystanie z tego ostatniego jest już niezalecane, w związku z czym powinieneś stosować wyłącznie atrybut id.

Dowiedz się więcej

Dobry projektant stron WWW nie powinien nigdy przestać poznawać nowych technologii, takich jak XML i XHTML. Oczywiście najlepszym miejscem, w którym można się z nimi zapoznać jest strona domowa konsorcjum W3C (*www.w3.org*). Warto zaznajomić się z zawartością następujących podstron:

- **HTML:** *www.w3.org/MarkUp*
- **XML:** *www.w3.org/XML*