1. Wstęp

XHTML (Extensible HyperText Markup Language) jest hybrydą pomiędzy HTML-em a XML-em, zaprojektowaną specjalnie dla internetowych urządzeń pokazujących WWW. XHTML jest napisany w XML-u, jest zatem jego aplikacją (podobnie jak XML jest aplikacją SGML-a). Jego wersja 1.0 datowana jest na 26 stycznia 2000, a obecna 1.1 na 31 maja 2001. Powstał z przetworzenia składni HTML 4 na standard XML. Dzięki temu dokumenty w nowym języku są w pełni zgodne z XML i mogą wykorzystywać wszystkie jego właściwości. Teoretycznie powinny także być lepiej interpretowane przez przeglądarki WWW. Jednym z powodów opracowania XHTML jest coraz większa dostępność i popularność alternatywnych (w stosunku do komputera PC) metod dostępu do Internetu. Konieczne jest zatem umożliwienie zachowania z nimi zgodności. Znacznie łatwiej jest także dla webmasterów zbliżyć się poprzez XHTML do standardu XML.

HTML, mimo swojej popularności, ma wiele wad zmniejszających jego przenośność. Przez lata dodawano do niego rozszerzenia powodujące „puchnięcie" przeglądarek internetowych. HTML dobrze opisuje wygląd dokumentu, ale nie najlepiej jego strukturę. Wprowadzenie XML poprawiło tę sytuację, ale niestety, nie wszystkie przeglądarki rozumieją ten standard. Wyjściem z tej trudnej sytuacji jest właśnie XHTML. Dla przeglądarki nie znającej XML, dokument XHTML-owy będzie zwykłym dokumentem napisanym w HTML. Bardziej zaawansowane programy wykorzystujące możliwości XML potraktują go jako dokument XML i będą mogły skorzystać z funkcji przez niego oferowanych. XHTML jest zatem pomostem ułatwiającym przejście z HTML na XML. Nie musimy pozbywać się zgodności z już istniejącymi.

Innym problemem, stojącym przed webmasterem, jest rosnąca liczba platform, na których można uruchamiać przeglądarki. Platformy te cechują się dużą różnorodnością. HTML można oglądać już nie tylko na ekranach monitorów, ale także w telewizorze, ciekłokrystalicznym ekranie palmtopa, a także korzystając z telefonu komórkowego. Każde z tych urządzeń ma inne możliwości przetwarzania HTML-a, a także wyposażone jest w odmienne urządzenia wejścia/wyjścia. Skrzynka WebTV będzie miała podłączoną myszkę lub joystick, ale może nie mieć klawiatury, z kolei przeglądarka w telefonie komórkowym ma bardzo ograniczone możliwości prezentacyjne, zarówno jeśli chodzi o rozdzielczość obrazu, jak i liczbę kolorów.

2. Moduły

Rozwiązaniem okazuje się modularyzacja standardu XHTML. Wszystkie możliwości HTML-a zostały podzielone na części zwane modułami.

XHTML został podzielony na następujące moduły:

1. Structure
2. Basic Text
3. Hypertext
4. List
5. Applet
6. Presentation
7. Edit
8. BDO
9. Forms
10. Tables
11. Image
12. Image Map
13. Intrinsic Events
14. Metainformation
15. Scripting
16. Stylesheet
17. Lin

Przeglądarka działająca na danej platformie może obsługiwać tylko niektóre z nich, ale za to obsługuje je w całości. Pozwala to tak przygotować stronę, aby nawet na mocno ograniczonych urządzeniach była ona wyświetlana poprawnie i czytelnie. Modułowa budowa XHTML-a pozwala na zdefiniowanie i dodawanie nowych znaczników i atrybutów do już istniejących. Tworzy to mechanizm do rozszerzania i rozbudowy XHMTL-a wychodząc naprzeciw przyszłym potrzebom użytkowników na pojawiających się nowych platformach sprzętowych. Modułowość XHTML będzie powodowała, iż przeglądarki internetowe w przyszłości będą dostarczały obszar roboczy, w którym twórcy stron internetowych będą mogli podłączyć (plug in) każdy rodzaj modułu przeznaczonego do wyświetlania konkretnej zawartości i konkretnego typu danych. Umożliwi się w ten sposób grupom specjalistów z danej dziedziny np. chemikom, lekarzom czy matematykom pracę nad rozwojem różnych profili używających standardowych elementów HTML-a rozszerzonych o elementy spełniające oczekiwania specjalistów. Tak będzie się rozwijał XHTML. W XHTML pola formularzy (form fields) są związane ze zdefiniowanym typem danych i mniej z konkretną prezentacją (presentation), co ułatwia wykorzystanie i przetwarzanie danych przez inne oprogramowanie, np. generowanie raportów, statystyk, itd.

Nowa specyfikacja XForms uczyni łatwiejszym dostosowanie do regionalnych różnic w walucie, numerach telefonów, określaniu daty, adresów pocztowych i umożliwi współdziałanie z wielką różnorodnością urządzeń zewnętrznych jak mikrofony, kamery, skanery.

Dwa główne powody dlaczego powinno się używać XHTML-a, to:

- możliwości rozszerzania. Dokumenty XML muszą być poprawnie sformułowane (be well formed), czyli elementy muszą być poprawnie zagnieżdżone. W HTML-u dodanie nowej grupy elementów wymagało zmiany całego DTD. W przypadku dokumentów XHTML wymagane jest tylko, aby nowe elementy były wewnętrznie zgodne i poprawnie sformułowane, aby zostały dodane do istniejącego DTD. Cecha ta w ogromny sposób ułatwia rozwój i integrację nowych zbiorów elementów,
- możliwość szerokiego zastosowania. Przewiduje się, iż będzie ciągle wzrastała popularność urządzeń nie będących komputerami (non-desktop devices) oferujących dostęp do Internetu, a co za tym idzie będzie wzrastała różnorodność platform sprzętowych. W większości przypadków urządzenia te nie są przystosowane do obsługi źle sformułowanych (ill-formed) dokumentów HTML, które są obsługiwane przez przeglądarki na tradycyjnych komputerach. Właściwie przeglądarki niedesktopowe, jeżeli nie otrzymają dobrze sformułowanych znaczników, to nie będą w stanie wyświetlić zawartości dokumentu.

3. Przykładowe użycie XHTML-a

Dokument XHTML nie różni się znacząco od dokumentu HTML poza różnicami, które zaznaczone są poniżej:

1. Deklaracja XML
2. Określa definicję typu dokumentu (DOCTYPE) na XHTML Transitional
3. Deklaracja jako dokumentu HTML i deklaracja XHTML namespaces
10. Znacznik obrazka jest zamknięty, cudzysłów przy wartości
12. Zamknięcie akapitu

```
1:  <?xml version="1.0" encoding="iso-8859-2"?>
2:  <!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.0 Transitional//EN"
    "DTD/xhtml1-transional.dtd">
3:  <html xmlns="http://www.w3.org/1999/xhtml">
4:  <head>
5:  <title>Przykład</title>
6:  </head>
7:  <body>
```

```
8:  <h1>Przykład</h1>
9:  <a href="http://lux.dmcs.p.lodz.pl">
10: <img src="obraz.jpg" height="600" width="800" border= "0"
    align="center" alt="obraz" /></a>
11: <p>Witam na mojej stronie.
12: </p>
13: </body>
14: </html>
```

Dokument XHTLM zaczynamy od:

```
<?xml version="1.0" encoding="iso-8859-2"?>
```

gdzie `xml` oznacza, że dokument wykorzystuje język XML, natomiast `encoding` oznacza domyślny sposób kodowania znaków (można zmienić). Następnie definiujemy typ dokumentu (jeden z trzech możliwych):

1. **`Strict`** **-** realizuje „ideał” proponowany w specyfikacji HTML 4.01: bez znaczników i atrybutów określanych jako **deprecated**, a na dodatek - bez stosowania ramek:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "DTD/xhtml1-
strict.dtd">
```

2. **`Transitional`** **-** dopuszcza wszystkie zdefiniowane elementy z wyjątkiem ramek:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-
transitional.dtd">
```

3. **`Frameset`** **-** dopuszcza wszystkie zdefiniowane elementy, także ramki (wersja przeznaczona wyłącznie dla strony definiującej układ ramek):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "DTD/xhtml1-
frameset.dtd">
```

Zwykle stosuje się typ **`Transitional`**, czasem też **`Frameset`** dla zestawów ramek. Konieczne jest także podanie ścieżki do dokumentu DTD zawierającego informacje o danym typie. Można wykorzystać dokumenty zawarte na stronie W3C, pod adresami:

```
http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd
http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd
```

Wygląda to następująco dla trzech kolejnych typów kodowania:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        " http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd ">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Dalej wstawiany jest znacznik:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

ze wskazaniem na lokalizację użytego standardu i język dokumentu oraz użytych w nim znaczników.

Wszystkie dokumenty XHTML muszą:

1. zawierać deklarację DOCTYPE pojawiającą się przed główną częścią dokumentu (root element) tzn. przed znacznikiem `<html>`,
2. określać jedną z trzech dostępnych DTD dla XHTML (strict, transitional, frameset),
3. posiadać część główną dokumentu określającą XHTML namespace, tzn. należy dodać znacznik XML namespace xmlns i wartości definiujące stronę jako dokument XHTML.

4. Czym różni się XHTML od HTML-a?

Mimo, że dokument XHTML jest podobny do dokumentu napisanego w HTML, to jednak musi przestrzegać kilku ograniczeń:

1. zakresy działania tagów nie mogą się nakładać. Mimo, że formalnie nie jest to zgodne z definicją HTML, to jednak wiele przeglądarek dopuszczało taki zapis. W XHTML jest to nielegalne:

   ```
   <p>tu jest trochę <em>tekstu.</p></em>          niepoprawnie
   <p>tu jest trochę <em>tekstu</em>.</p>          poprawnie
   ```

2. wszystkie tagi i nazwy argumentów muszą być pisane małymi literami. Wynika to z tego, że XML rozróżnia wielkie i małe litery, a XHTML jest właśnie napisany w XML-u:

```
<IMG SRC="abc.gif" WIDTH="123" HEIGHT="10">     niepoprawnie
<Img src="abc.gif" Width="123" hEIght="10">     niepoprawnie
<img src="abc.gif" width="123" height="10" />   poprawnie
```

3. wszystkie niepuste elementy muszą mieć tagi kończące:

```
<p>This is not XHTML complaint code             niepoprawnie
<p>This is XHTML complaint code</p>             poprawnie
<option>More Options                            niepoprawnie
<option>More Options</option>                   poprawnie
<base />                                        poprawnie
<meta />                                        poprawnie
<dt>first list</dt>                             poprawnie
```

4. wartości atrybutów muszą być ujęte w cudzysłowie:

```
<table width=100 bgcolor=#000000>               niepoprawnie

<table width="100" bgcolor="#0000000">          poprawnie
```

5. niedopuszczalne jest skracanie atrybutów:

```
HTML              <td nowrap>text</td>
XHTML             <td nowrap="nowrap">text</td>
HTML              <dl compact>
XHTML             <dl compact="compact">
HTML              <ul compact>
XHTML             <ul compact="compact">
HTML              <option .. selected>
XHTML             <option .. selected="selected">
HTML              <td nowrap>text</td>
XHTML             <td nowrap="nowrap">text</td>
HTML              <input type="radio" .. checked>
XHTML             <input type="radio" .. checked="checked" />
HTML              <input type="checkbox" .. checked>
XHTML             <input type="checkbox" .. checked="checked" />
```

6. puste elementy muszą mieć tagi kończące albo muszą być kończone znakami:

```
<br><hr>                    niepoprawnie

<br /><hr />                poprawnie
```

Spacja między znacznikiem a slash-em na końcu jest dodawana, aby uniknąć problemów z przeglądarkami.

7. wewnątrz wartości atrybutów początkowe i końcowe spacje są usuwane, zaś wielokrotne spacje zastępowane pojedynczym znakiem:

```
<a href= "abc.gif">          niepoprawnie
<a href="abc.gif">           poprawnie
```

8. wewnątrz znacznika `<script>` i `<style>` znaki < lub & zostaną rozwinięte do < i &. Aby temu zapobiec należy otoczyć je sekcją CDATA jak niżej:

```
<script>
<![CDATA[
...
]]>
</script>
```

9. W przypadku użycia zakotwiczeń (anchor) należy użyć zarówno `id` i `name` w znaczniku:

```
<a name="cyberplex">...</a>                          niepoprawnie
<a id="cyberplex" name="cyberplex">...</a>           poprawnie
```

Dokumenty XHTML, jako zgodne z większością przeglądarek obecnych na rynku, mogą być oznaczane sygnaturą MIME: text/html.

5. Usunięte znaczniki

Język XHTML zakłada usunięcie jednych znaczników i przejęcie ich funkcji przez inne lub zastąpienie ich arkuszem stylów.

Stary sposób (już niedozwolony)	Nowy sposób
`<applet>`	`<object>`
`<basefont size="x" color="y" face="z">`	`<body style="font-size:x; color:y; font-family:z">`
`<center>`	`<div style="text-align:center">`
`<dir>`	`<ul>`
`<font size="x" color="y" face="z">`	`<span style="font-size:x; color:y; font-family:z">`
`<isindex>`	`<input type="text" />`

<listing>	<pre>
<menu>	
<plaintext>	<pre>
<s> lub <strike>	
<u>	
<xmp>	<pre>

6. Usunięte atrybuty

Podobnie jak w przypadku niektórych tagów, również nie wszystkie atrybuty zostają przeniesione z HTML 4.01. Oto te, których nie należy już używać:

Stary sposób (już niedozwolony)	Nowy sposób
accesskey=	brak odpowiednika
align= (można go jeszcze używać w elementach związanych z tabelą)	różnie w zależności od znacznika
alink=	<style type="text/css"> a:active {color:x}</style>
background=	style="background-image: url('nazwa')"
bgcolor=	style="background-color:x"
clear=	style="clear:x"
color=	style="color:x"
compact=	brak odpowiednika
language=	<script type="text/javascript">
link=	<style type="text/css"> a:link {color:x}</style>
start=	brak odpowiednika
tabindex=	brak odpowiednika
text=	style="color:x"
vlink=	<style type="text/css"> a:visited {color:x}</style>

Znaczniki wykorzystujące atrubut **align**:

- typ dokumentu Strict:

col, colgroup, tbody, td, tfoot, th, thead, tr
W przypadku wszystkich znaczników, atrybut **align** może przyjmować następujące wartości: left, center, right, justify, char

- typ dokumentu `Transitional`:

applet	top, middle, bottom, left, right
caption	top, bottom, left, right
col, colgroup	left, center, right, justify, char
div, h1, h2, h3, h4, h5, h6	left, center, right, justify
Hr	left, center, right
iframe, img, input	top, middle, bottom, left, right
legend	top, bottom, left, right
object	top, middle, bottom, left, right
P	left, center, right, justify
Table	left, center, right
Tbody, td, tfoot, th, thead, tr	left, center, right, justify, char

- typ dokumentu `Transitional`:

Applet	top, middle, bottom, left, right
Caption	top, bottom, left, right
Col	left, center, right, justify, char
Colgroup	left, center, right, justify, char
div, h1, h2, h3, h4, h5, h6	left, center, right, justify
hr	left, center, right
iframe, iframe, img, input	top, middle, bottom, left, right
Legend	top, bottom, left, right
object	top, middle, bottom, left, right
P	left, center, right, justify
Table	left, center, right
Tbody, td, tfoot, th, thead, tr	left, center, right, justify, char

Znacznik **`align`** nie jest zalecany. Proponuje się wykorzystywać w stylach (`text-align, vertical-align`):

przykład:

```
<h3 align="right">Tekst</h3>                    nie zalecany sposób

<h3 style="text-align: center">Tekst</h3>       realizacja na stylach
```

Znaczniki wykorzystujące atrybut **`height`**:

- typ dokumentu `Strict`:

`img, object`

- typ dokumentu `Fransitional`:

`applet, iframe, img, object, th, td`

- typ dokumentu Frameset:

`applet, iframe, img, object, th, td`

Znaczniki wykorzystujące atrubut **width**:

- typ dokumentu `Strict`:

```
col, colgroup, img, object, table
```

- typ dokumentu `Transitional`:

```
applet, col, colgroup, hr, iframe, img, object, pre, table, td, th
```

- typ dokumentu `Frameset`:

```
applet, col, colgroup, hr, iframe, img, object, pre, table, td, th
```

Atrybut **height** nie zalecany jest w elementach **TD**, **TH** i **APPLET**. Proponuje się zastąpienie stylem:

```
<img src="../p/sm.gif" height=”24” />
```

```
<td style=”height: 24”>Tekst</td>
```

Atrybut **width** nie zalecany jest w elementach **HR**, **TD**, **TH**, **APPLET**, **PRE**. Proponuje się zastąpienie stylem:

```
<img src="../p/sm.gif" width=”24” />
```

```
<td style=”width: 24”>Tekst</td>
```

Wszystkie materiały zostały znalezione w Internecie.

Adresy stron WWW:

http://www.w3.org

http://webmaster.pckurier.pl/2000/marzec/wieczorek/xhtml.html

http://www.kt.agh.edu.pl/~romanow/stud/przyd.html

http://nethut.pl/artykul.php/t/104

http://zvon.org/xxl/xhtmlReference/Output/index.html