

Projektujesz w tabelkach? OBUDŹ SIĘ!

Pomóż rozwijać się sieci w dobrym kierunku. Dave Shea ([mezzoblue](#)) został prywatnie poproszony o wyjaśnienie projektantowi starej daty, który chciałby zagłębić się w standardy sieciowe, CSS, XML i XHTML, od czego powinien zacząć.

Wnikliwą odpowiedź postanowił udostępnić szerokiemu gronu, a ja postanowiłem ją przetłumaczyć. Bo to bardzo oczywiste, przystępne i precyzyjne kompendium dla webdesignera, który chce wkroczyć na właściwą ścieżkę.

Oddaję głos.

Stój! Zanim cokolwiek zrobisz, dla własnego dobra powinienes na wstępie uświadomić sobie 2 sprawy: a) nauka zajmie trochę czasu, b) w jej trakcie obgryziesz resztkę paznokci.

Ale nie jesteś sam. Wielu z nas przeszło to samo, a jesteś w o tyle lepszej sytuacji, że pojawia się coraz więcej źródeł, które ułatwią Ci pracę. Starzy wyjadacze musieli opracowywać wszelkie sztuczki i techniki, które dziś są dla nas oczywiste. Szczęściarze, którzy pojawili się później (np. ja) mogą korzystać z wiedzy wypracowanej potem i łzami.

Kiedy w końcu Twoje umiejętności posługiwania się standardami przyćmią stare metody projektowania za pomocą tabelki, spojrzysz wstecz i zrozumiesz jak wielce sensowne jest projektowanie przy użyciu kaskadowych arkuszy stylów. Z drugiej strony, dzisiejsze sposoby tworzenia układów stron w css mogą wydawać się średnio sensowne (zwłaszcza, że CSS2 oferuje znacznie lepsze możliwości niż te, które dzisiaj mamy do dyspozycji). Jednak za brak wsparcia dla standardów należy winić producenta najbardziej popularnej przeglądarki.

Znów tu zabrnąłem. Przykro mi, ale kolejną rzeczą, do której będziesz musiał się przyzwyczaić jest nieokietznana nienawiść do Internet Explorera. Zanim zrozumiesz, konieczne będzie trochę praktyki, ale w krótkim czasie będziesz dokładnie wiedział, o czym mowa.

Przejdźmy więc do rzeczy. Najpierw kup "[Projektowanie serwisów WWW. Standardy sieciowe](#)". Nie myśl, po prostu idź kupić tę pozycję. Już ją masz? W porządku, teraz nie kaź jej się kurzyć. Przeczytaj. Wszystko, o czym tu napiszę, jest szczegółowo przedstawione w tej książce. Jest tam manifest (DLACZEGO miałbyś to robić?) i tutorial (JAK to zrobić?). Potrzebujesz jej.

Teraz tak - pierwszą rzeczą jest przestawienie się na XHTML. Nie ważne, czy wybierasz HTML 4.01 czy XHTML 1.0 Strict (są powody dla wyboru jednego i drugiego, na razie się nimi nie przejmuj, a nawet ignoruj je do czasu, kiedy będziesz gotów na zmianę myślenia) - wszystko zaczyna się od DOCTYPE.

Informacja dla przeglądarki o tym, jak sformatowany jest Twój dokument, pozwoli uniknąć niepożądanych błędów w wyświetlaniu strony, które, w przeciwnym wypadku, doprowadziłyby Cię do szału. Pomyśl o tym w ten sposób: jeśli chcę lecieć do Chicago, muszę określić miejsce docelowe, kupując bilet. Pewnie, że fajnie byłoby zaryzykować i wylądować w nieznanym miejscu, ale niekoniecznie praktycznie. DOCTYPE zapewnia podróż we właściwym kierunku, bez przypadkowych wyskoków do Wiednia.

Następny cel: porządny markup. Bułka z masłem. Atrybuty zawsze zawieraj w cudzostowie (np. ``, i `<input type="text" />`). Uważaj na poprawne zagnieżdżanie elementów, zamykaj je w kolejności. Każdy element wymaga otwarcia i zamknięcia.

Dygresja: gdzieś po drodze "tagi" stały się "elementami". Taka sama składnia, inna teoria. Nazywaj je jak chcesz, od teraz nazywam je elementami - może zawsze nimi były? Nie wiem. Nikt mi tego nigdy nie wytłumaczył.

Jeśli stosujesz HTML 4.01, nie musisz zamykać elementów takich jak `
`, `<hr>`, `<input>`. W XHTML jest to konieczne. Tutaj stosujemy inną składnię, którą bez problemów zignorują starsze przeglądarki - po prostu dodaj spację i slash na końcu. I tak np. `
` zamienia się na `
`.

Jest jeszcze jeden istotny szczegół, jeśli chodzi o atrybuty w XHTML - zawsze muszą mieć wartość, nawet jeśli ma ona niewielki sens. Przykład: `<input type="radio" checked="checked" />`. W HTML 4.01 wystarczy samo `checked`, ale XHTML wymaga uzupełnienia.

I w końcu - XHTML piszemy małymi literami. HTML nie rozróżnia ich wielkości, natomiast XHTML używa składni XML, w której ma to znaczenie.

To wszystko w kwestii markupu! Odpręż się i odpocznij, ponieważ to dopiero pierwszy krok.

Teraz, kiedy już potrafisz pisać poprawnie w HTML/XHTML, spróbuj przepuścić swój kod przez [validator](#) W3. Jeśli wszystko napisałeś poprawnie i logicznie, oczom twym ukaże się potwierdzenie zgodności ze standardami. Naucz się kochać tę niebiesko-żółtą wiadomość, może się ona okazać Twoim najlepszym przyjacielem.

Dlaczego walidacja jest ważna, a nawet stosowna? Ponieważ kiepsko napisany kod może dać kompletnie nieoczekiwane efekty - uzależniasz się od sposobu interpretacji błędów przez różne przeglądarki. I chociaż radzą sobie one z tym nieźle, poleganie na nich jest złą praktyką. To właśnie doprowadziło do wojen przeglądarek w pierwszej kolejności (Microsoft mógł rywalizować z Netscape w czasach jego popularności w 1995, ponieważ IE został zaprojektowany w taki sposób, by ignorować te same błędy co Netscape).

Walidacja pomaga wyłapać błędy w kodzie, co z kolei zapewnia lepsze wyświetlanie strony. Pierwszą rzeczą, którą robię przy problemach z układem strony, to przetestowanie kodu. Ty też powinieneś.

Tak, to rzeczywiście może być frustrujące, gdy po pierwszym teście widzisz 78 błędów. Niestety, choć walidator pomaga, nie jest perfekcyjny. Prowadzą go ochotnicy, którzy robią co mogą, jednak czasami błędy są tak potrzebne, jak poezja Vagonów. Na szczęście błędy się nakładają, co oznacza, że jak znajdziesz jeden niedomknięty element paragrafu <p>, ilość istniejących błędów może się zmniejszyć o kilkanaście. Czyli wygląda źle, a często nie jest najgorzej.

Analizując kod poprzez walidację, kroczysz ścieżką prawa. Jak to bywa i z rzeczywistymi prawami, stosujesz się tylko do surowo wytyczonych reguł, nie wiedząc w ogóle po co.

Następnym krokiem będzie oczyszczenie kodu z elementów układu i prezentacji, które w najnowszych typach dokumentów nie są już tolerowane, i przeniesienie ich do oddzielnego pliku. To jest właśnie słynne oddzielenie treści od prezentacji. Właśnie teraz wkracza CSS.

Wygląda to tak: tekst jest Twoją treścią. Treść jest w porządku, ale bez wskazówek o jej strukturze (w którą wliczają się np. spacje, nagłówki i listy), jest zwyczajnie kupą bezużytecznego tekstu. Struktura to dodatkowa warstwa, która zamienia tę zbieraninę wyrazów w logiczne grupy i porządkuje w sposób, który pozwala rozróżnić indywidualne elementy na stronie. Wygląd może być częściowo podyktowany przez strukturę (np. z reguły tekst głównego nagłówka strony będzie większy niż pozostały). Ale na tym kończy się jej wpływ na wygląd.

Czas więc na warstwę prezentacji. Tutaj określamy, że nagłówek pierwszego stopnia ma być czerwony, kursywny, i mieć 150% rozmiaru głównej czcionki. Kod prezentacji jest dodatkową warstwą zawartą nad kodem struktury, która ma wpływ na jej wygląd w bardzo wyrafinowany sposób. Za pomocą CSS możemy przekształcić bardzo prosty markup w wizualne cacko. Sprawdź [css Zen Garden](#).

Więc jak najlepiej rozpocząć oddzielanie struktury od prezentacji? Weź kawałek kodu HTML i uznaj go za swojego wroga, urzeczywistnienie najgorszych intencji wobec sieci. Czas zniszczyć wszystkie atrybuty typu bgcolor czy <center>. Mały quiz:

Które atrybuty w poniższych przykładach usunąłbyś jako ślady warstwy prezentacji?

1. <center><h1>Moja pierwsza strona.</h1></center>
2. <table border="0" cellpadding="0" cellspacing="0">
3. <body bgcolor="#ffffff" topmargin="0" leftmargin="0" marginwidth="0" marginheight="0">
4. <td bgcolor="#ffffff" valign="top" align="center"><p>Oczywiście dzień dobry...</p></td>

Znasz już odpowiedzi? Świetnie. Porównaj z listą poniżej. Oto właściwe elementy strukturalne bez śladu formatowania:

1. <h1>Moja pierwsza strona.</h1>
2. <table>
3. <body>
4. <td><p>Oczywiście dzień dobry...</p></td>

Tylko to? Tylko to.

Chociaż żadna specyfikacja tak **nie nakazuje**, dalszym etapem tego rozdzielania powinno być używanie właściwych elementów we właściwych celach. Używanie tabeli do tworzenia układu strony jest więc, z definicji, błędnym zastosowaniem tabeli. We wcześniejszym przykładzie mogłoby się nawet okazać konieczne usunięcie elementu <table>, ale nie mogliśmy być tego pewni bez kontekstu. Tabele nie wychodzą z użycia, wciąż mogą być bardzo przydatne. Lecz powinno się ich używać zgodnie z przeznaczeniem - do wyświetlania danych tabularnych.

Udało się w końcu usunąć prezentację ze struktury. Co teraz? Pozbyliśmy się całkiem linearnych informacji, a gdzie te wspaniałe wizualne możliwości, które nam obiecano?

Wróćmy do przykładu [Zen Garden](#). Widzisz te fajne projekty? Widzisz jak bardzo są różne? Aby to docenić musisz wiedzieć, że u podłoża każdego z tych projektów leży jeden i ten sam plik XHTML. Naprawdę - [zobacz](#).

Posiadanie takiej "ogółoconej" podstawy jest w rzeczywistości bardzo korzystne. Zapewne zauważyłeś, że ten niesformatowany dokument przypomina wyglądem niezbyt piękną sieć z 1994r. I, z kilkoma wyjątkami, taki właśnie jest. Jest tak stary jak internet. Element <h2> istnieje już od czasów przeglądarki Mosaic, która, czego się pewnie właśnie domyślasz, i dzisiaj nadzwyczaj poprawnie wyświetli naszą stronę. Spróbuj to powiedzieć o tabelkowych monstrach z lat 90-tych.

To oczywiście nie koniec korzyści. Dostępność dla osób o specjalnych potrzebach uzyskujemy za darmo, optymalizacja dla wyszukiwarek jest samoczynna, możemy zredukować koszty hostingu (wraz z rozwojem sieci), i tak dalej, i tak dalej. Jeffrey Veen napisał artykuł o [Zaletach standardów sieciowych w biznesie](#), Roger Johansson rozwinął wątek w publikacji "[Projektowanie w standardach sieciowych](#)".

Bhang: Po polsku polecam przeczytanie choćby tych pozycji:

- [Zalety stosowania standardów sieciowych dla Twoich Czytelników, Klientów i Ciebie!](#)
- [Handlowe korzyści stosowania standardów sieciowych](#)
- [W głąb dostępności](#)

CSS jest dziś wspierane przez wszystkie większe przeglądarki, a w sieci dostępne są niezliczone źródła do nauki składni, podstaw układu i zaawansowanej teorii. Wskażę Ci kilka z nich (bhang: źródła po angielsku): WestCiv oferuje [darmowy kurs CSS](#), który pomoże zacząć i rozpędzić się w tej dziedzinie. Andrew Fernandez zgromadził [spora listę źródeł](#), które powinny Ci pomóc, niezależnie od poziomu wiedzy. Eric Meyer napisał kilka książek, które powinieneś mieć, np. serię [Eric Meyer on CSS](#) i [More Eric Meyer on CSS](#) (naprawdę, tak właśnie są zatytułowane). Inne pozycje to wydana w O'Reilly (a w polskim Helionie) [CSS. Kaskadowe arkusze stylów. Przewodnik encyklopedyczny](#). Warta uwagi jest również książka Molly Holzschlag [CSS: The Designer's Edge](#), i Chrisa Schmitta [Designing CSS Web Pages](#).

Wnikanie w zalety i wady stosowania CSS i tworzenia układów mogłyby zająć o wiele więcej miejsca, niż już wykorzystałem i pewnej finansowej motywacji. Zakończmy więc tutaj, biorąc pod uwagę fakt, że dotrwanie do tego miejsca stanowiło z Twojej strony nie lada wyczyn.

Podłączcie się - to najlepsza rada, jaką mogę dać każdemu, kto chce zacząć stosować standardy. Poprzez ciągłe doskonalenie się i dzielenie wiedzą, rozwijamy się jako społeczność. Wielu z nas aktywnie działa jako twórcy tych standardów i coraz więcej zacznie działać w najbliższych latach. Do dyspozycji mamy globalną sieć komunikacji, upewnij się, że z niej korzystasz.

A jeśli zbłądzisz, jesteśmy tu, gotowi Ci pomóc. To zdarza się każdemu!

Michał Mokrzycki - [artykuł](#) opublikowałem za zgodą Dave'a Shea...