CSS

Cz.2

Menu w CSS

Za pomocą odpowiedniej struktury HTML i dobranego stylu CSS można w prosty i szybki sposób zrobić ciekawe menu.

Wykorzystane zostaną listy nieuporządkowane. Dlaczego?

Ponieważ jest to poprawne semantycznie. Poprawność semantyczna to stosowanie znaczników zgodnie z ich przeznaczeniem, czyli np. do tworzenia menu wykorzystuje się listę, a nie tabelkę.

Lista to zbiór następujących po sobie pozycji, a więc zastosowanie jej do prezentacji pozycji menu jest jak najbardziej prawidłowe. Domyślne ułożenie punktów listy będzie wygodne w przeglądarkach tekstowych, które nie obsługują CSS. Będzie to również wygodne dla niewidomych, którzy do czytania stron internetowych używają syntezatorów mowy.

Tego nie rób…

**Do tworzenia menu na pewno nie nadaje się znacznik
, ponieważ wtedy zmiana wyglądu za pomocą CSS byłaby możliwa tylko w niewielkim stopniu - nie można by np. stworzyć menu poziomego.**

Nie jest również dobrym pomysłem zastąpienie listy zwykłymi elementami ... , ponieważ w przeciwieństwie do listy, nie niosą one ze sobą żadnego znaczenia semantycznego.

Struktura menu (HTML)

XHTML z odnośnikami, który będzie zupełnie pozbawiony formatowania:

```xhtml
<ul>
<li><a href="czcionki.html">Czcionki</a></li>
<li><a href="tekst.html">Tekst</a></li>
<li><a href="tlo.html">Tło</a></li>
<li><a href="marginesy.html">Marginesy</a></li>
<li><a href="obramowanie.html">Obramowanie</a></li>
</ul>
```

Efekt bez CSS dość mizerny:

- Czcionki
- Tekst
- Tło
- Marginesy
- Obramowanie

Teraz wystarczy dołożyć CSS:

```css
ul, ul li {
        display: block;
        list-style: none;
        margin: 0;
        padding: 0;
}

ul {
        width: 200px;
}

ul a:link, ul a:visited {
        display: block;
        width: 186px;
        text-decoration: none;
        background-color: #ccc;
        color: #000;
        padding: 5px;
        border: 2px outset #ccc;
}

ul a:hover {
        border-style: inset;
        padding: 7px 3px 3px 7px;
}
```

Efekt->

Szablony stron oparte na DIV

Wiesz już zapewne, dlaczego rozwiązanie oparte o CSS jest najkorzystniejszym wyborem przy tworzeniu menu nawigacyjnego. Dlaczego zatem ograniczać się tylko do menu, kiedy można przecież zyskać jeszcze więcej, projektując całą stronę w ten sam sposób.

Wspominałem, że nie powinno budować się szablonu strony w oparciu np. o tabele.

Najlepsze do tego celu są bloki DIV, trzeba je jedynie odpowiednie ułożyć i dodać do nich kod CSS by wyglądały w taki sposób jak tego oczekujemy.

Kilka przykładów:

Stały szablon (ang. fixed layout) charakteryzuje się odgórnie ustaloną szerokością w pikselach. Jeśli użytkownik zmieni rozdzielczość ekranu lub rozmiar okna przeglądarki, taki szablon będzie zajmował inną powierzchnię szerokości ekranu. Aby zapewnić komfortowe przeglądanie strony, szerokość szablonu nie powinna być szersza od najmniejszej przewidzianej rozdzielczości ekranu, z której mają korzystać czytelnicy serwisu - w przeciwnym razie w oknie przeglądarki pojawi się poziomy suwak i przeczytanie całej linijki tekstu może wymagać ciągłego przewijania w prawo i z powrotem. W wyższych rozdzielczościach ekranu pojawi się puste miejsce po lewej lub/i po prawej stronie właściwego szablonu.

Dwie kolumny:


```css
html, body {
        background-color: #fff;
        color: #000;
        margin: 0;
        padding: 0;
        text-align: center;
}

#top {
        width: 780px;
        margin-left: auto;
        margin-right: auto;
        text-align: left;
}

#NAGLOWEK {
        background-color: #888;
}

#MENU {
        width: 150px;
        float: left;
        overflow: hidden;
        background-color: #ccc;
}

#TRESC {
        width: 630px;
        float: left;
        overflow: hidden;
        background-color: #fff;
}

#STOPKA {
        clear: both;
        width: 100%;
        background-color: #888;
}
```

Wyjaśnienie:

Pierwszą regułą stylów w arkuszu CSS jest ustalenie domyślnego koloru tła oraz kolor tekstu w całym dokumencie. Zawsze trzeba o tym pamiętać, ponieważ użytkownik może zmienić domyślne kolory w swoim systemie operacyjnym, a wtedy przypadkowo możemy uzyskać dość osobliwy efekt kolorystyczny - np. czarny tekst na czarnym tle. Nigdy nie zakładaj, że domyślny kolor tekstu to czarny, a kolor tła - biały!

W tym samym miejscu usuwamy wszystkie marginesy strony. Analogicznie postępujemy z marginesem wewnętrznym, aby zabezpieczyć się przed możliwą odmienną interpretacją w niektórych przeglądarkach. Wyzerowanie marginesu jest konieczne, aby uzyskać maksymalną dostępną przestrzeń w poziomie.

```css
html, body {
        background-color: #fff;
        color: #000;
        margin: 0;
        padding: 0;
        text-align: center;
}
```

Wyjaśnienie:

Ustalamy odpowiednią szerokość szablonu, tzn. taką, aby zmieściła się w najniższej planowanej rozdzielczości ekranu - w tym przypadku 800x600. Musimy przewidzieć 20px miejsca dla suwaka do przewijania treści, dlatego ostatecznie, przy zerowych marginesach strony, ustalamy szerokość szablonu na 780px.

```css
#top {
        width: 780px;
        margin-left: auto;
        margin-right: auto;
        text-align: left;
}
```

Wyjaśnienie:

Dla bloku nagłówka po prostu przypisujemy odrębny kolor tła, aby był lepiej widoczny na załączonych przykładach.

```css
#NAGLOWEK {
        background-color: #888;
}
```

Wyjaśnienie:

Cała "magia" dzieje się dla bloku treści strony i menu nawigacyjnego. Normalnie wszystkie bloki szablonu były ułożone jeden pod drugim. Aby ułożyć wspomniane kolumny pionowo, użyliśmy własności float: left, wykorzystywanej zwykle przy oblewaniu elementów (np. ilustracji) tekstem. Kilka następujących po sobie bloków z określonym oblewaniem, ustawia się obok siebie - o ile mieszczą się w dostępnej szerokości ich rodzica.

Suma szerokość wszystkich kolumn nie może przekroczyć 780px, ponieważ jako najmniejszą dopuszczalną rozdzielczość ekranu założyliśmy 800x600 (od wartości 800 odejmujemy szerokość pionowego suwaka).

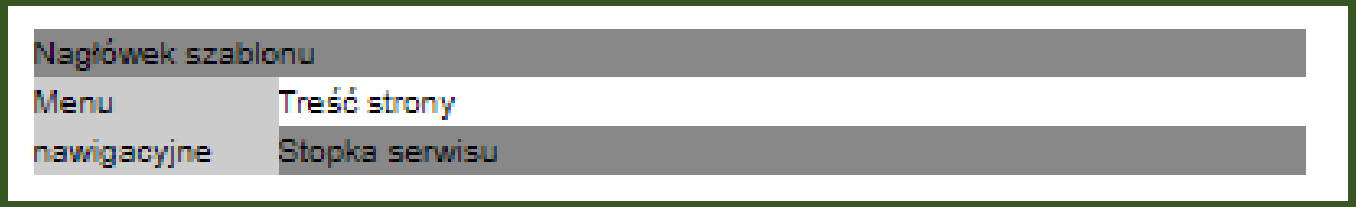
Oblewane bloki kolumn posiadają również własność przepełnienia overflow: hidden. Jest to zabezpieczenie na wypadek, gdyby w ich zawartości znalazły się elementy szersze niż mogą pomieścić. W takim przypadku, w zależności od przeglądarki, zbyt szerokie elementy mogłyby zachodzić na sąsiednie bloki, zasłaniając ich zawartość albo blok taki mógłby zostać przeniesiony poniżej wcześniejszego, co zupełnie zepsułoby szablon.

```
#MENU {
        width: 150px;
        float: left;
        overflow: hidden;
        background-color: #ccc;
}

#TRESC {
        width: 630px;
        float: left;
        overflow: hidden;
        background-color: #fff;
}
```

Wyjaśnienie:

Stopce serwisu nadajemy własność przylegania clear: both, ponieważ bez tego szablon mógłby wyglądać następująco:



Efekt ten wynika z ustalonego oblewania (float: left) dla środkowych bloków szablonu, które próbują wyświetlić się obok (w tym przypadku po lewej stronie) pozostałych elementów, które w kodzie źródłowym są umieszczone pod nimi. Deklaracja width: 100% została dodana tylko ze względu na błąd Internet Explorera 6.0 (w MSIE 7.0 wszystko jest w porządku), objawiający się wyświetleniem koloru tła stopki od dolnych krawędzi ustawionych powyżej oblewanych bloków, a tekstu stopki prawidłowo - poniżej nich.

Więcej przykładów w Internecie…☺

Płynny szablon

Płynny szablon (ang. liquid layout) charakteryzuje się zmianą swoich poziomych proporcji przy zmianie rozmiaru okna przeglądarki lub rozdzielczości ekranu. Najczęściej w każdych warunkach zajmuje on całą dostępną szerokość w oknie. Jest on raczej rzadziej stosowany, ze względu na niemożliwe do przewidzenia ułożenie elementów treści. Poza tym tekst w zbyt długich linijkach zwykle gorzej się czyta, ponieważ trudniej przenieść wzrok z końca wiersza na początek następnego. Czasami jednak może być wygodny, np. kiedy zawiera jakieś szerokie elementy, które mogą nie zmieścić się w ustalonej szerokości dla stałego szablonu (czyli zwykle 780px minus szerokość kolumny menu i ewentualnie dodatkowych informacji).

Przykłady w Internecie.

KONIEC cz.2