pasja-informatyki.pl

Programowanie webowe

Mirosław Zelent

Odcinek #1

ten, w którym udaje się nam zrozumieć składnię
kodu witryny oraz zainkludować CSS i JS

Spis treści

Rozpoczynamy pracę z kodem!

Odcinek wstępny kursu za nami – pora najwyższa wziąć się do pracy z kodem źródłowym! Jeżeli poprzednia (zerowa) część kursu została przez Ciebie przegapiona, to PDF do pobrania, test ze znajomości technologii webowych oraz premierowe video serii znajdziesz tutaj:

http://pasja-informatyki.pl/programowanie-webowe/technologie-tworzenia-witryn/

Przejdźmy do napisania pierwszej strony internetowej w języku HTML – od tego niewątpliwie warto rozpocząć swoją przygodę z programowaniem webowym. Zrozumieć znaczniki – tak można by w dwóch słowach podsumować cel pierwszej lekcji w dowolnym kursie tworzenia stron internetowych.

Jednak cel naszej dzisiejszej pracy jest inny: chcemy zrozumieć przeznaczenie każdej jednej wstawionej do kodu linii, i to niejednokrotnie w dość szerokim kontekście. Porozmawiamy więc o ewolucji standardów sieciowych, o optymalizowaniu strony dla wyszukiwarek (z ang. SEO - Search Engine Optimization), zwrócimy uwagę na rolę umiejętności miękkich w zdobywaniu wiedzy, dowiemy się jak poprawnie kodować polskie ogonki, oraz podepniemy do pliku HTML pierwszy arkusz stylów CSS oraz skrypt JS.

Będziemy niepokornie ciekawi, obsesyjnie zainteresowani, szukając frywolnie odpowiedzi na pytanie: „Dlaczego właśnie tak?”. Zdarzy nam się nawet bezwstydnie podejrzeć kod HTML innych programistów webowych (jak rasowy paparazzi zbadamy źródła popularnych portali) ale i zajrzymy do anglojęzycznych, wyczerpujących źródeł wiedzy. Niech ta lekcja przekona Cię, jak wciągającym, ciekawym i wielowymiarowym zajęciem jest tworzenie stron internetowych!

Wybór edytora (IDE), konfiguracja Notepad++

Czy po odcinku zerowym dokonałeś/aś już wyboru edytora kodu albo IDE? Po pierwsze wyjaśnijmy różnicę pomiędzy edytorem kodu a IDE.

Edytor kodu – program komputerowy umożliwiający nam wygodne pisanie linii kodu, zazwyczaj podświetlający składnię, numerujący kolejne linie, umożliwiający zaawansowaną edycję źródła (wyszukiwanie i zamiana tekstu, system podpowiedzi, konfiguracja kodowania znaków w pliku etc.). Często też edytory pozwalają nam zastosować tzw. tematy (skórki) – mamy do dyspozycji wiele różnych szablonów kolorystycznych, a jako że ludzie posiadają różne gusta i widzą

przeróżne misie, ułatwia to spersonalizowanie procesu pisania własnego kodu. Przykłady edytorów: Notepad++, Sublime Text, Atom, Brackets.

IDE (ang. Integrated Development Environment) – jest to tak zwane: *zintegrowane środowisko programistyczne*. Zatem to już nie tylko edytor kodu, ale cały zestaw narzędzi pozwalający między innymi: uruchamiać napisane źródła, wygodnie je debugować (czyli wyszukiwać w nich błędy), testować, jak i dbać o strukturę i odpowiedni porządek realizacji całego projektu. Przykłady IDE: Eclipse, NetBeans, PHPStorm.

Uwaga: czasami spotkasz się z próbami uczynienia z edytora IDE poprzez np. dołączenie do niego odpowiedniej ilości wtyczek (pluginów) – stąd też ta różnica pomiędzy edytorem a IDE często bywa płynna. Podobnie jeśli nie używasz zaawansowanych opcji IDE, to niczym nie różni się to od używania edytora. Jeżeli ktoś uwielbia jakiś edytor, to często do np. debugowania użyje IDE, ale napisze większość swojego kodu w ukochanym edytorze. Nie jest też tak, iż pisanie w IDE automatycznie czyni z Ciebie lepszego programistę, podobnie jak chwalenie się tym przed ludźmi używającymi edytorów – jest to zachowanie podobne do postępowania osób, które przeczytawszy Grę o Tron chwalą się tym nałogowo wszystkim, którzy jedynie obejrzeli serial.

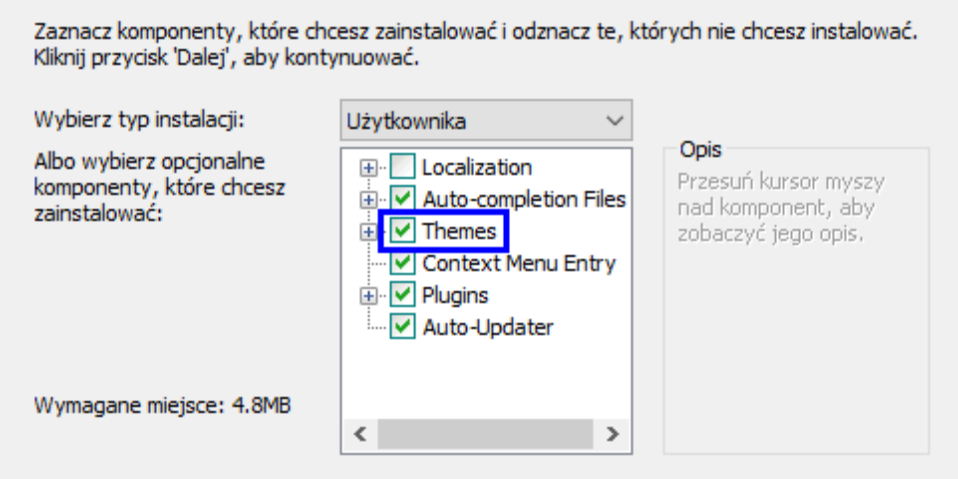
Co zatem wybrać? Na poziomie technikum, a już na pewno w pierwszych kontaktach z tworzeniem stron w ogóle, wystarczy jakikolwiek edytor podświetlający składnię i posiadający ciemne tło (choć nic nie stoi na przeszkodzie, aby od razu było to IDE). Reguła jest prosta: dokonaj wyboru na podstawie subiektywnego wrażenia „jak mi się pisze" w danym programie. Nie poświęcaj jednak zbyt wiele uwagi bawieniu się kolorkami i tematami w edytorach – pracuj z kodem, to na początku najważniejsze! Do świadomego wyboru swojego środowiska powrócisz na etapie szlifowania własnych umiejętności (po opanowaniu podstaw tworzenia witryn).

Na egzaminie E.14 na stanowisku dostępny będzie edytor kodu, najczęściej Notepad++ (proponowany przez CKE), aczkolwiek to asystent techniczny przygotowujący egzamin w danej szkole decyduje ostatecznie jakie edytory zainstalować (jeżeli zależy Ci na obecności jakiegoś konkretnego edytora na stanowisku, to koniecznie powiedz mu o tym odpowiednio wcześnie).

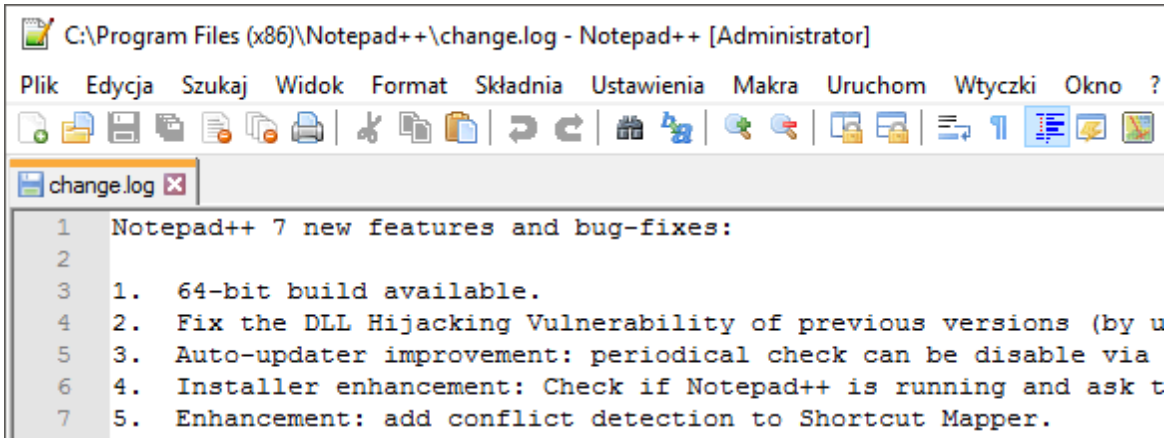
Jeżeli nie masz pomysłu co wybrać do nauki w tym kursie, to możesz na czas naszej wspólnej pracy korzystać z Notepada++ (to jego używam na nagraniach video w niniejszym kursie).

Edytor można pobrać stąd: https://notepad-plus-plus.org/ (najlepiej wybierz pakiet ze słowem „installer" – wówczas będzie zawierał gotowy instalator).

Podczas instalacji warto upewnić się, że zainstalowane zostaną tematy (themes):



Świeżo po instalacji Notepad++ ma ustawiony domyślny temat z białym tłem:



W trosce o Twoje oczy, warto ustawić ciemny temat w edytorze, na przykład skórkę o nazwie Obsidian – w tym celu wybieramy z menu Ustawienia -> Konfigurator stylów i na liście wyboru wskazujemy właśnie ten temat. Oczywiście styl Obsidian to jedynie propozycja, sam wybierz przyjemny dla swojego oka temat.

Gorąco zachęcam także do przestawienia interfejsu edytora na język angielski: Ustawienia -> Preferencje i z listy „Język" wybrać „English". Taki zabieg ułatwia (na dłuższą metę) przyswojenie przez Twój umysł wielu popularnych zwrotów angielskich związanych bezpośrednio z webdeveloperką (idea „otoczenia się" językiem, którego się uczymy).

Jeżeli zaś masz ochotę używać innej czcionki niż Courier New, w konfiguratorze stylów zaznacz opcję: „Używaj globalnego kroju czcionki". Dobrym wyborem może okazać się krój Lucida Console.

Linia po linii, w telegraficznym skrócie

Poniżej znajdziesz możliwie najbardziej skrótowe wyjaśnienie do czego służą poszczególne linie stworzonej przez nas w odcinku templatki HTML. Ponieważ szczegółowy opis znajdziesz w materiale video, tutaj przedstawiłem skrócone informacje w formie podręcznej „ściągawki". Cała templatka prezentuje się następująco:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="utf-8">
    <title>Książki science-fiction, które musisz
    przeczytać!</title>
    <meta name="description" content="Lista najlepszych
    książek z zakresu fantastyki, science-fiction wartych
    przeczytania! Fantastyka naukowa - TOP Lista Wszech
    Czasów">
    <meta name="keywords" content="książki, science, fiction,
    fantastyka, sf">
    <meta name="author" content="Trevor Philips Industries">

    <meta http-equiv="X-Ua-Compatible"
    content="IE=edge,chrome=1">

    <link rel="stylesheet" href="main.css">
    <script src="code.js"></script>

</head>

<body>

</body>
</html>
```

Rozpocznijmy analizę od doctype'u:

```
<!DOCTYPE html>
```

Informujemy przeglądarkę, iż niniejszy dokument należy potraktować jako zapisany w standardzie HTML 5. Oczywiście przeglądarka posiada wiele mechanizmów kompatybilności wstecznej ze standardami HTML 4.01 i XHTML (więc jeżeli zapiszemy coś wg starszej metody nie będzie wielkiego problemu), niemniej jednak taka deklaracja powinna się znaleźć jako pierwsza w kodzie Twojej strony. Starsze deklaracje były dużo dłuższe, bo zawierały tzw. wpis DTD (ang. document type definition) – przykłady starszych deklaracji znajdziesz na przykład na W3Schools: http://www.w3schools.com/tags/tag_doctype.asp

To teraz pora sprawić, aby **polskie ogonki wyświetlały się poprawnie**. Potrzebujemy dopilnować obecności dwóch zapisów oraz poprawnie ustawić kodowanie pliku HTML:

1. Ustawienie polskiego języka witryny

```
<html lang="pl">
```

2. Użycie zestawu znaków utf-8 (lub ewentualnie iso-8859-2) – tag meta charset powinien być pierwszym tagiem wstawionym do sekcji head.

```
<meta charset="utf-8">
```

Istnieje także dłuższa wersja tagu, którą można stosować zamiennie z wersją powyższą – więcej szczegółów znajdziesz tutaj: http://stackoverflow.com/questions/4696499/meta-charset-utf-8-vs-meta-http-equiv-content-type

3. To samo kodowanie (zestaw znaków, czyli charset) ustawiamy dla naszego dokumentu HTML – kodowanie można sprawdzić / zmienić w edytorze (IDE). W przypadku Notepada++ zaglądamy do menu Format (lub w wersji angielskiej interfejsu do opcji Encoding).

To teraz pora określić tytuł (znaczniki title):

```
<title>
   Książki science-fiction, które musisz przeczytać!
</title>
```

Tytuł podstrony jest bardzo ważny w kontekście SEO – zwróć szczególną uwagę na jego długość (wykorzystaj całe dostępne miejsce) oraz zawartość (tytuł powinien zawierać kluczowe frazy, na które pozycjonujemy witrynę).

Kolejny ważny w kontekście SEO znacznik meta – opis strony w wyszukiwarce:

```
<meta name="description" content="Lista najlepszych książek z
zakresu fantastyki, science-fiction wartych przeczytania!
Fantastyka naukowa - TOP Lista Wszech Czasów">
```

Do dyspozycji mamy około 155-160 znaków. Opis powinien składać się zarówno z kluczowych fraz, jak również bezpośrednio zwracać się do internauty / zainteresować go właśnie naszą witryną.

Pora na słowa kluczowe (meta keywords):

```
<meta name="keywords" content="książki, science, fiction, fantastyka, sf">
```

Co prawda robot Google'a obecnie praktycznie ignoruje tę sekcję (z powodu nadużyć internautów w przeszłości), jednak po dziś dzień umieszczamy ją w kodzie witryny wpisując kilka najbardziej kluczowych, reprezentatywnych dla witryny fraz. Robimy to pomimo traktowania po macoszemu tego znacznika – ot, bez przesadnej czułości i płonnych nadziei wstawiamy go siłą tradycji nie licząc jednak na zbyt wiele benefitów w zamian.

Określenie autorstwa witryny:

```
<meta name="author" content="Trevor Philips Industries">
```

Tag opcjonalny, ale wciąż możliwy do umieszczenia. W praktyce dużo lepiej podpisać się na stronie linkiem do własnej witryny (znacznik <a>) – wówczas przynajmniej zyskujemy kolejny link prowadzący do nas w rezultatach wyszukiwania Google (a to już wymierna korzyść).

Warto jeszcze zatroszczyć się o poprawne wyświetlenie witryny w starszych wersjach IE:

```
<meta http-equiv="X-Ua-Compatible" content="IE=edge, chrome=1">
```

Atrybut content powinien mieć wartość IE=edge, z opcjonalnym chrome=1. Można też określić konkretną wersję standardów IE, wpisując na przykład IE=10. Pełną dyskusję znajdziesz tutaj – zachęcam do lektury:

http://stackoverflow.com/questions/6771258/what-does-meta-http-equiv-x-ua-compatible-content-ie-edge-do

Podpięcie (zainkluowanie) arkusza stylów CSS:

```
<link rel="stylesheet" href="main.css">
```

Atrybut: type=”text/css” jest opcjonalny, a nawet zbędny, aczkolwiek jego wstawienie nie jest błędem. Zwróć zawsze szczególną uwagę na podanie właściwej, względnej (czyli nie zawierającej litery dysku) ścieżki do pliku. Nie zapomnij też wykonać od razu prostego testu podpięcia – zmień roboczo tło strony na ciemne przekonując się, że arkusz stylów rzeczywiście został dołączony prawidłowo do dokumentu HTML.

Podpięcie (zainkluowanie) skryptu JS:

```
<script src="code.js"></script>
```

Atrybut: type=”text/javascript” jest opcjonalny, a nawet zbędny w HTML 5, aczkolwiek jego wstawienie nie jest błędem. Więcej szczegółów na ten temat znajdziesz w tej dyskusji: http://stackoverflow.com/questions/4243577/which-is-better-script-type-text-javascript-script-or-script-scr Ponownie, nie zapomnij o prostym teście prawidłowego podpięcia skryptu – wystarczy jeden mały alert w pliku JS aby przekonać się, że kod JS rzeczywiście jest interpretowany przez przeglądarkę. Uwaga: w samym pliku z rozszerzeniem JS nie trzeba już pisać tagów <script>.

Na koniec wspomnijmy jeszcze o dyskusji na temat domykania tagów w HTML 5 w porównaniu do XHTML i HTML 4.01 – czy należy kończyć pojedyncze znaczniki znakiem / czy nie? Czyli czy powinno się zapisać:

```
<meta charset="utf-8">
```

czy jednak:

```
<meta charset="utf-8" />
```

Otóż zgodnie ze specyfikacją HTML 5 w tagach pojedynczych obowiązuje brak kończącego znaku / (czyli wersja pierwsza), aczkolwiek zapis znany z XHTML (czyli domknięcie tagu – wersja druga) nie powoduje błędu w przeglądarce – będzie ona wiedziała, ze chodzi o domknięcie znacznika. Wielu ludzi nadal używa wersji XHTML-owej, jest to kolejny przykład tego jak płynne, powolne i kompatybilne wstecz jest wprowadzanie nowych standardów sieciowych. Więcej informacji na ten temat znajdziesz tutaj: http://stackoverflow.com/questions/7366344/do-we-still-need-end-slashes-in-html5

Umiejętności miękkie

Dyskusje na temat optymalnych zapisów w HTML często bywają „gorące”, gdyż ludzie zbyt często angażują w nie swoje ego. Płynność, powolna dynamika i kompatybilność wsteczna standardów sieciowych także nie ułatwiają ludziom podjęcia jednoznacznych konkluzji. W sieci znajdziesz bardzo wiele opinii na ten temat i bardzo wielu „bojowników o sprawę”. Ty zachowaj jednak chłodny dystans pozbawiony emocji, przede wszystkim skupiając się na wielokryterialnej optymalizacji **swoich własnych** kodów źródłowych.

Wiedza nie powinna prowadzić do chęci wywyższania się i bezpardonowego krytykowania innych. Temperament, ostre słowa i natychmiastowe osądzanie (często podszyte podświadomą chęcią poczucia się ważnym) to domena ludzi młodych i życiowo niedoświadczonych. Życzę

Tobie, aby wiedza dała Ci sporo dystansu oraz empatyczego zrozumienia innych punktów widzenia. A przede wszystkim aby pozwoliła Tobie stawać się codziennie nieco lepszym niż się było wczoraj.

Pierwsza zasada w książce Dale'a Carnegiego *How to Win Friends and Influence People* brzmi: *czysta krytyka nie służy nikomu*. Po szerszy kontekst odsyłam do książki (nie chcę okradać autora z jego dzieła), jednak od razu warto świadomie zwrócić uwagę na niebezpieczeństwo użycia wiedzy do wywyższania się – nie karm tego wilka, a dzięki temu szybciej zrozumiesz dynamikę relacji międzyludzkich. Rozwinięte umiejętności miękkie są niezwykle ważne także w pracy programisty webowego (w relacjach z klientami jak i współpracownikami).