



pasja-informatyki.pl

Programowanie webowe

Mirosław Zelent

Odcinek #0

ten, w którym poznajemy technologie
webowe oraz naturę procesu nauki

Spis treści

| | |
|---|----|
| Prolog | 3 |
| Czy programowanie webowe jest dla mnie? | 5 |
| Architektura klient-serwer | 7 |
| Front-end | 8 |
| HTML..... | 8 |
| CSS..... | 9 |
| JavaScript..... | 10 |
| Back-end | 11 |
| PHP, SQL, MySQL | 12 |
| Drzewo technologii webowych | 14 |
| Gałąź front-endowa | 15 |
| Gałąź back-endowa..... | 16 |
| Umiejętności fundamentalne..... | 17 |
| Przydatne linki..... | 18 |

Prolog

Tworzenie stron internetowych to zajęcie, które niesamowicie rozwija na wielu różnych płaszczyznach. Jest to zajęcie kreatywne, angażujące i dające wiele uporu w realizacji swoich planów. Uczy logicznego myślenia, rozwija zdolność analizy i cierpliwość. Jest mi niezwykle miło, że Ty również inwestujesz w siebie przystępując do przepracowania niniejszego kursu!

Zanim przejdziemy do rzeczywistej pracy z kodem, warto zaznajomić się ze ścieżką rozwoju programisty webowego oraz zrozumieć przeznaczenie pięciu podstawowych technologii webowych (HTML, CSS, JavaScript, PHP, MySQL). Nie może też zabraknąć dokładnego omówienia klasycznej dychotomii „Front-end vs. Back-end”, jak również kilku słów odpowiedzi na bardzo ważne, choć często niedoceniane pytanie: „Jak się uczyć?”.

Kurs niniejszy stworzyłem dla uczniów techników informatycznych, którzy przygotowują się do egzaminu zawodowego. Sprawy uczniów technikum są mi szczególnie bliskie, gdyż sam od sześciu lat, w różnym wymiarze czasowym, zajmuję się dydaktyką programowania w publicznej szkole. Na przestrzeni tych lat spotkałem wielu niesamowitych młodych ludzi, którzy dziś prowadzą już własne, dorosłe życie. Pomimo przeróżnych problemów i ograniczeń (tak uczniów, organizacji szkoły, jak i moich) wytrwale pracowaliśmy nad byciem każdego dnia lepszymi niż byliśmy wczoraj. Móc towarzyszyć Wam w tej wytrwałej pracy, czasami pomagać i kierować, ale głównie nie przeszkadzać w Waszym naturalnym rozwoju – to był prawdziwy zaszczyt!

Od trzech lat realizuję także dydaktykę w internecie, głównie poprzez kanał YouTube, bloga miroslawzelent.pl i trenera egzaminacyjny egzamin-informatyk.pl. Na dzień dzisiejszy łączny czas oglądania filmów umieszczonych na kanale Pasja informatyki wynosi 126 lat i 206 dni. Często powtarzam na filmach, że „szanuję Wasz czas” – może wydawać się, że to jedynie powtarzane niczym mantra cliché, ale ja rzeczywiście w to wierzę. Nasz czas na ziemi jest ograniczony i nawet przy całym swoim wysiłku nie jesteśmy w stanie dodać do długości swojego życia nawet jednej sekundy.

Jedyne na co mamy wpływ, to wybór w jaki sposób realizować swoją własną inicjatywę. Internet pełen jest ludzi mocno roszczeniowych. Powoli, iskra po iskrze to mnie wypala, ale taka jest natura świata, który zawsze jest dualistyczny, słodko-gorzki, kreacyjno-destrukcyjny. Wielu ludzi krytykować będzie ten kurs czy nawet mnie osobiście, tylko dlatego że informacje zawarte w tej serii nie będą przeznaczone dla nich – chcieliby zobaczyć na kanale kolejne nowe serie, nauczyć się czegoś przydatnego nieco dalej na ścieżce rozwoju programisty webowego.

Ja jednak uważam, iż początkowy etap pracy z kodem (czyli rozpoczęcie nauki programowania w ogóle) jest najbardziej newralgicznym momentem na drodze rozwoju w branży IT, często stanowiącym o być albo nie być w tym zawodzie. Jest też etapem, który łatwo dla kogoś zepsuć, kładąc zbyt wielki ciężar na jego barki, albo odwrotnie – nie przykładając się należycie do rozpalenia ognia pasji programowania z pierwszych drobnych iskier. I dlatego poświęcam temu szczególnie wiele uwagi na kanale Pasja informatyki, niwelując początkowe trudności dla wielu adeptów fascynującej sztuki programowania. Nie oznacza to jednak, iż po przepracowaniu z pietyzmem fundamentów, nie mam zamiaru realizować serii trudniejszych.

Dlatego tym bardziej doceniam wsparcie ludzi, którzy zamiast pisać tuż po premierze komentarze typu "A może zrobisz serię" czy też „Kiedy kolejny odcinek ” wolą wesprzeć naszą ciężką pracę łapkowaniem, pozytywnym komentarzem dla statystyk albo nawet patronowaniem nas w serwisie Patronite. Dziękuję również za tony korespondencji, w której opisałyście Wasze historie oraz to jak wykonana przeze mnie i Damiana praca dydaktyczna okazała się dla Was pomocna i cenna. Takie maile motywują nas do dalszej pracy - regularnej, małymi krokami. Ciężko jest znaleźć czas na YouTube, ale takie wiadomości pokazują, że warto to robić! Dziękujemy i mamy nadzieję dostarczać Wam dobrej jakości contentu tak długo, jak nie wypali się w nas pasja do nauczania i wyjątkowo czasochłonnej wideoedycji.

Niniejszym oddaję w Wasze przeglądarki serię tutoriali „Programowanie webowe” – jestem pewien, że praca jaką wspólnie w tym kursie wykonamy przyniesie w długim okresie czasu zadziwiające rezultaty, jednak pamiętaj – najwięcej zależy zawsze od Ciebie! Przejdziesz przez wszystkie fazy procesu nauki – pojawi się entuzjazm, lekkość i łatwość przyswajania wiedzy, ale pojawią się także fazy plateau, stagnacji, jak również zniechęcenia, awersji czy rezygnacji. Nie martw się – taka jest naturalna kolej rzeczy, tak zawsze wygląda trening jakiejkolwiek umiejętności! Miej dla siebie cierpliwość, nie porównuj się z innymi, staraj się być po każdej sesji po prostu odrobinę lepsz(a) niż byłeś wczoraj.

Programowanie ma niezwykle długą krzywą uczenia się. Mistrzostwo rozpoczyna się, kiedy przestajesz na siłę szukać finału tej drogi, i skupiasz się na jakości samej podróży, na jakości stawiania kroków. Tobie też się uda poznać naturę tego zmagania ze sobą. Ty także zrozumiesz jak trudna i wymagająca, ale jednocześnie piękna jest, codzienna praca nad sobą w branży IT.

Kurs to nie tylko tutoriale wideo, oddajemy do Waszej dyspozycji także materiały PDF, kody źródłowe, linki warte odwiedzenia oraz testy do samodzielnego przepracowania. Dla osiągnięcia pełni rezultatów, korzystaj ze wszystkich tych źródeł wiedzy i ćwiczeń. Powodzenia!

Czy programowanie webowe jest dla mnie?

Paradygmat działania klasycznej aplikacji internetowej nie zmienił się od lat – jest oparty na tzw. architekturze klient-serwer i protokole HTTP. Wszędzie widać to „HT”: HTML, HTTP – o co tutaj chodzi? HT oznacza Hypertext, czyli hiperłącza (linki) – od lat sześćdziesiątych dwudziestego wieku taki mamy pomysł na działanie stron internetowych: kliknięcie w hiperłącze potrafi przenieść nas do kolejnej podstrony, a czasem także na inny serwer, do kolejnego miejsca w sieci. Porównanie do pajęczyny nasuwa się tu samo. Pierwsze strony główne stanowiły po prostu spis tekstowych linków – stąd tradycja nazywania dokumentu głównego nazwą: index.html (index = skorowidz, lista tematów, indeks pojęć, taki jak na końcu papierowego podręcznika).

HTTP to akronim oznaczający Hypertext Transfer Protocol, zaś HTML to Hypertext Markup Language. Markup Language, czyli „język znaczników”. I rzeczywiście, kiedy wejdziemy na dowolną stronę internetową i wybierzemy w przeglądarce opcję „Wyświetl źródło strony” to naszym oczom ukaże się tekstowy opis witryny, niczym matriksowy kod stanowiący drugie dno oglądanej naszymi oczami rzeczywistości:



```
1 <!DOCTYPE html> <html> <head> <meta charset="utf-8" /> <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
<meta name="msapplication-task" content="name=Wiadomości;action-uri=http://wiadomosci.onet.pl;icon-uri=http://ocdn.eu/p
engine/4710e72d24bea6cac94c786a87b6526f/faviconn2.ico"/> <meta name="msapplication-task" content="name=Biznes;action-uri=h
engine/4710e72d24bea6cac94c786a87b6526f/faviconn2.ico"/> <meta name="msapplication-task" content="name=Sport;action-uri=h
```

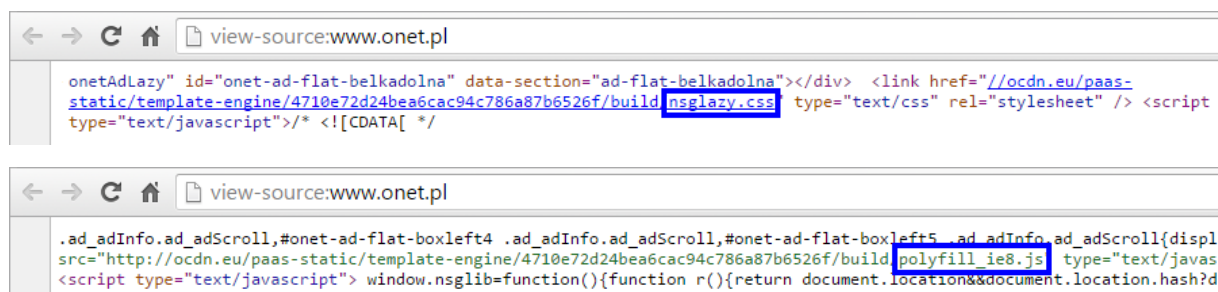
Znacznik po znaczniku (czy jak mówią niektórzy: tag po tagu), ten kod determinuje strukturę strony, określa sposób jej wyświetlania, definiuje wszystkie elementy interfejsu, które zapełnią wirtualne płótno karty przeglądarki. Cholernie to ciekawe dla ludzi lubiących pogłównkować, w tym zwłaszcza dla introwertyków. Widząc taki kod, człowiek (jako system otwarty), zadaje sobie najważniejsze na świecie pytanie: „Dlaczego?” oraz „Co te zapisy oznaczają?”. Ciekawość nas swędzi, ciekawość nas boli – potrzebujemy wiedzieć i rozumieć. I dlatego programowanie przyciąga nas, ludzi ciekawych świata, jak magnes. Łączy dziecięcą fascynację tajemnicą z dorosłym pragmatyzmem i zimną jak sztylet z lodu logiką.

Podjęcie tego intelektualnego wyzwania: „chcę wiedzieć, chcę zrozumieć”, oznacza też wysiłek, chwile zwątpienia, walkę z własnymi słabościami, z brakiem cierpliwości, z chęcią rzucenia tego w cholerę. Jednak jest coś godnego w podjęciu tej próby, istnieje w tej decyzji wiele hartu ducha. Przypomina to odpowiedź Neo na pytania Agenta Smitha (film „Matrix”): „Dlaczego walczysz? Dlaczego mimo tylu upadków wstajesz i kontynuujesz ten proces?” – „Bo chcę” – odpowiada Neo, albo też odczytując jego imię jako anagram: „One” - ten wybrany.

Coś w tym jest – tak jak wiedźmińską próbę traw przetrwało średnio 3 na 10 adeptów, tak przygodę z programowaniem przechodzą jedynie nieliczni, ci wybrani. Większość ludzi porzuca naukę po wyczerpaniu się początkowego entuzjazmu, który naturalnie zawsze towarzyszy byciu w czymś początkującym. I nie ma w tym nic złego – w życiu testujemy wiele różnych dziedzin, wiele różnych przedmiotów zainteresowań, ścieżek rozwoju, dróg życiowych, możliwych pasji. Ostatecznie, programowanie może nie okazać się dla Ciebie tym, czym chcesz się w życiu zajmować zawodowo i to jest jak najbardziej OK. Być może też jesteś tutaj tylko po to, aby po prostu zdać egzamin – i to też jest w porządku, uwierz mi – okaże się to całkiem nietrudne.

Zachęcam Cię tylko do jednego: podejź do kursu z otwartą głową, z krytycznym myśleniem, oraz ze szczerością wobec siebie. Poznasz naukę ścisłą jaką jest programowanie, a także poobserwujesz siebie w trakcie nauki – to bezcenne doświadczenia. Jednak dopiero po wejściu nieco głębiej w tę dziedzinę zdołasz odpowiedzieć sobie na pytanie: czy ja to lubię (na tym poziomie) i czy sprawia mi to radość? Jak również: Czy na pewno na co dzień wygląda to tak jak myślałem(a)m na początku że wygląda? Cokolwiek postanowisz – czas spędzony na kursie programowania nauczy cierpliwości, uporu i analityki – słowem: nie będzie to czas stracony!

A wracając do okna z widokiem kodu HTML w przeglądarce – po bliższym przyjrzeniu się tym enigmatycznym zapisom, odnajdujemy w kodzie witryny także linki do plików .css oraz .js:



```
view-source:www.onet.pl

onetAdLazy" id="onet-ad-flat-belkadolna" data-section="ad-flat-belkadolna"></div> <link href="//ocdn.eu/paas-
static/template-engine/4710e72d24bea6cac94c786a87b6526f/build/nsglazy.css" type="text/css" rel="stylesheet" /> <script
type="text/javascript"> /*  */

view-source:www.onet.pl

.ad_adInfo.ad_adScroll,#onet-ad-flat-boxleft4 .ad_adInfo.ad_adScroll,#onet-ad-flat-boxleft5 .ad_adInfo.ad_adScroll{displ
src="http://ocdn.eu/paas-static/template-engine/4710e72d24bea6cac94c786a87b6526f/build/polyfill_ie8.js" type="text/javas
&lt;script type="text/javascript"&gt; window.nsglib=function(){function r(){return document.location&amp;&amp;document.location.hash?d</pre></div><div data-bbox="113 665 884 729" data-label="Text"><p>które to pliki także można bez problemu otworzyć i zajrzeć do całej ich zawartości. Jednak plików (czy też w ogóle pojedynczych linii) języków PHP i SQL próżno nam szukać w kodzie. Jak to jest, że kod HTML, CSS i JavaScript jest jawny, a PHP i MySQL ukryły się przed naszym okiem?</p></div><div data-bbox="113 747 884 906" data-label="Text"><p>Odpowiedź jest wbrew pozorom banalna i logiczna: otóż do plików z kodem PHP czy MySQL nie możemy jako zwykły internauta zajrzeć, gdyż pliki te znajdują się jedynie na serwerze, na którym umieszczony jest portal onet.pl. Tylko autorzy portalu mają dostęp do swojego kodu PHP i stworzonych przez siebie zapytań SQL (jak i własnej bazy danych MySQL). Zatem aby dobrze zrozumieć naturę tworzenia witryn internetowych, trzeba nam najpierw z uwagą prześledzić komunikację, która zachodzi pomiędzy klientem chcącym wejść na stronę www, a serwerem wysyłającym w odpowiedzi żadaną witrynę.</p></div>
```

Architektura klient-serwer

Kiedy wpisujemy w przeglądarce adres strony internetowej (np. `http://facebook.com`), to nasz lokalny komputer (klient) zwraca się do serwera facebookowego z tzw. żądaniem HTTP (ang. request) udostępnienia nam plików tejże witryny. Komunikacja odbywa się za pośrednictwem protokołu HTTP (bądź czasami szyfrowanego HTTPS), po porcie nr 80, który jest standardowym portem przeglądarki internetowej. Konkretny adres IP serwera uzyskiwany jest dzięki pośredniczącemu w nawiązaniu połączenia serwerowi DNS (usługa DNS zamienia łatwy do zapamiętania adres: „facebook.com” na adres IP serwera – w przypadku: 69.171.230.68

Jeżeli mamy do czynienia ze zwykłą, statyczną stroną internetową prezentującą taką samą zawartość wszystkim użytkownikom, to odpowiedź serwera ograniczy się do prostego dostarczenia nam plików potrzebnych do wyrenderowania strony przez naszą przeglądarkę. Tymi plikami są właśnie kody źródłowe HTML, CSS i JavaScript. Te źródła są jawne, gdyż siłą rzeczy muszą trafić na nasz komputer lokalny. To procesor naszej maszyny (a konkretnie proces przeglądarki internetowej) zajmie się wyrenderowaniem wyglądu strony na podstawie przysłanych z serwera plików z rozszerzeniami: .html, .css, .js.



Taki przebieg komunikacji sieciowej przyczynił się do powstania nowej definicji, ot zwykłej etykiety, jaką określamy te technologie webowe, których kod źródłowy każdy internauta może podejrzeć na dowolnej witrynie internetowej. Tak narodziła się nazwa: Front-end.

Front-end

Tym mianem określa się technologie webowe, których kody źródłowe wykonywane są przez procesor po stronie klienta (czyli w praktyce przez przeglądarkę internetową lokalnego komputera). Jako że pliki te muszą na nasz lokalny komputer trafić z serwera przed ich wykonaniem, to rzecz jasna do tych kodów można zupełnie otwarcie zajrzeć – są one, jak to mówimy: jawne. Do front-endowych technologii zaliczymy przede wszystkim: HTML, CSS i JavaScript. Przy czym przeznaczenie każdej z tych front-endowych technologii webowych jest nieco inne – szczegóły znajdziesz poniżej.

HTML

Jest to język opisowy, w którym przy pomocy znaczników (tagów) określamy co zawiera dana podstrona – mogą to być hiperłącza, obrazki, pola edycyjne, przyciski, tabele, listy numerowane, pojemniki na zawartość, akapity tekstu, nagłówki itd. Przykładowe elementy zdefiniowane na stronie głównej Facebooka przedstawiono na ilustracji poniżej, wraz z zaznaczeniem jak wygląda znacznik generujący konkretny komponent witryny w kodzie źródłowym:

The image shows a screenshot of the Facebook registration page with several HTML tags highlighted in orange boxes and connected to their corresponding UI elements by lines. The highlighted elements and their corresponding HTML tags are:

- Facebook logo:** `<div> ... </div>`
- Registration form input fields:**
 - Adres e-mail lub numer telefonu: `<input type="text">`
 - Hasło: `<input type="password">`
 - Numer telefonu komórkowego lub e-mail: `<input type="text">`
 - Podaj ponownie numer telefonu komórkowego...: `<input type="text">`
 - Nowe hasło: `<input type="password">`
- Registration button:** `<input type="submit">`
- Profile picture placeholder:** ``

The page also includes a header with the Facebook logo, a login section with fields for email/phone and password, and a registration section with fields for phone/email, password, and date of birth. The registration button is labeled "Rejestracja".

Nie przejmuj się, jeżeli nie rozpoznajesz jeszcze poszczególnych znaczników – przyjdzie na to pora już niebawem! Oprócz zawartości strony, w HTML określamy także istotne parametry

witryny – tytuł i opis w rezultatach wyszukiwania Google, język i zestaw znaków właściwy dla danego kraju (u nas poprawnie należy zakodować polskie ogonki: ą, ę, ś, ć, ź, ł itd).

Uwaga – znaczna część autorzytetów i autorów podręczników uważa, iż HTML nie jest językiem programowania – to głównie dlatego iż nie potrafi on podejmować decyzji czy wykonywać pętli, instrukcji wyboru etc. Stąd przyjęto się uważać, iż jest on jedynie tzw. językiem opisowym.

CSS

Są to tak zwane kaskadowe arkusze stylów (ang. Cascading Style Sheets). Służą one do opisania wyglądu elementów witryny, zdefiniowanych uprzednio w HTML. W arkuszach stylów odnajdziemy zapisy w postaci: właściwość, dwukropek, wartość:



Nie przejmuj się, jeżeli nie rozumiesz jeszcze zapisu koloru #47639E - wszystko w swoim czasie! Jedno możemy przyznać na pewno - idea prostego określania wyglądu elementu przy pomocy zestawu atrybutów wydaje się intuicyjna i prosta. Ta belka jest niebieska, a te pola edycyjne używają czcionki Arial – naprawdę trudno w życiu o coś prostszego! Oczywiście używanie arkuszy stylów CSS oznaczać będzie także zrozumienie czym są tzw. klasy, identyfikatory, potomkowie, hovers, itd. ale na to jeszcze nie pora na tym etapie. Kodem CSS możemy także wpływać na położenie elementów i wzajemne relacje między nimi – czeka na nas zatem w kursie cały szereg atrakcji!

A dlaczego są to kaskadowe arkusze stylów? Kaskada to rodzaj wodospadu, mającego budowę schodkową. Jest to wyraźna aluzja do potężnego mechanizmu „schodkowego” właściwości, który niejednokrotnie zaoszczędzi nam wiele pracy, jednak teraz nie zaprzataj tym sobie głowy – po prostu teraz jesteś świadomy skąd się wzięła litera C w akronimie CSS.

Nasz dociekliwy umysł zadaje sobie jeszcze jedno nurtujące pytanie: skąd wzięła się ta (dziwaczna na pozór) idea rozdzielenia zawartości witryny od warstwy jej wyglądu? Otóż ma to duży sens praktyczny – odczujemy to na przykład w momencie, gdy jednym wpisem w arkuszu CSS uda nam się zmienić kolor wszystkich przycisków w witrynie. Stało się to możliwe, gdyż wszystkie podstrony w serwisie używały tego samego arkusza stylów, podpiętego do każdej z nich. To potężny i genialny w swojej prostocie mechanizm, naturalna ewolucja od czasów gdy wygląd określało się w HTML, na przykład z wielokrotnym użyciem tagów . Na szczęście te czasy bezpowrotnie minęły.

JavaScript

Jest to pełnoprawny, skryptowy język programowania, w którym możemy stosować cały repertuar klasycznych konstrukcji językowych (instrukcje warunkowe, pętle, zmienne, tablice, instrukcje wyboru, własne funkcje, klasy itd.). JavaScript (nie mylić z językiem Java, to co innego) to potężny front-endowy kombajn, za pomocą którego możemy tworzyć wyjątkowe od strony designu i interfejsu witryny.

Na samym początku swojej przygody z webdeveloperką, JS używamy najczęściej do ulepszenia interfejsu strony, wzbogacając ją o dodatkowe funkcjonalności, niedostępne w HTML czy CSS. Skrypty JS pozwolą nam tworzyć efektowne slidery, animowane galerie zdjęć, wyskakujące panele z nawigacją, interaktywne menu, zegary, animacje itd.

JavaScript umożliwia (po załadowaniu się strony do przeglądarki) podmianę wybranego fragmentu kodu HTML czy CSS, bez potrzeby ponownej komunikacji z serwerem. Na ekranie może zmienić się slajd, zegar będzie kontynuować odliczanie co sekundę, galeria fotek z wakacji pokaże nam widok pełnego zdjęcia po kliknięciu na jego miniaturkę, a panel z nawigacją w animowany sposób rozwinie się przed nami ujawniając dodatkowe możliwości interakcji.

JavaScript potrafi podejmować decyzje w zależności od zaistniałych okoliczności, stąd często w kontekście JS mówi się o programowaniu zdarzeniowym. Takim zdarzeniem (ang. event) może być upływ zadanej ilości czasu czy kliknięcie czegoś. Zadaniem programisty jest okodować odpowiednią obsługę takiego zdarzenia.

Wyobraźmy sobie YouTube'a – co innego musi się zdarzyć kiedy na przykład:

- klikniemy przycisk Pause na pasku narzędziowym odtwarzania filmu,
- damy łapkę w górę pod filmem spod znaku Pasja informatyki,
- prześlemy komentarz o treści: „Pierwszy!” albo „Darude – Sandstorm”,
- zasubskrybujemy kanał HowToBasic,
- klikniemy miniaturkę nowego teledysku Gangu Albanii,
- zmienimy rozmiar okna przeglądarki, znacząco je zawężając

O tak, najlepsze witryny w sieci zdecydowanie są interaktywne i żywo reagują na nasze poczynania. I dlatego właśnie potrzebujemy potężnego front-endowego kombajnu, który weźmie na siebie obsługę tego wszystkiego, co zrobić można lokalnie. Cały mechanizm responsywności stron też jest podszyty JavaScriptem. intuicyjne w obsłudze - do tego dążymy jako webdeveloperzy. I całe szczęście, że to właśnie JavaScript i pochodna JS biblioteka jQuery wyparły z rynku ciężką i toporną technologię Flash.

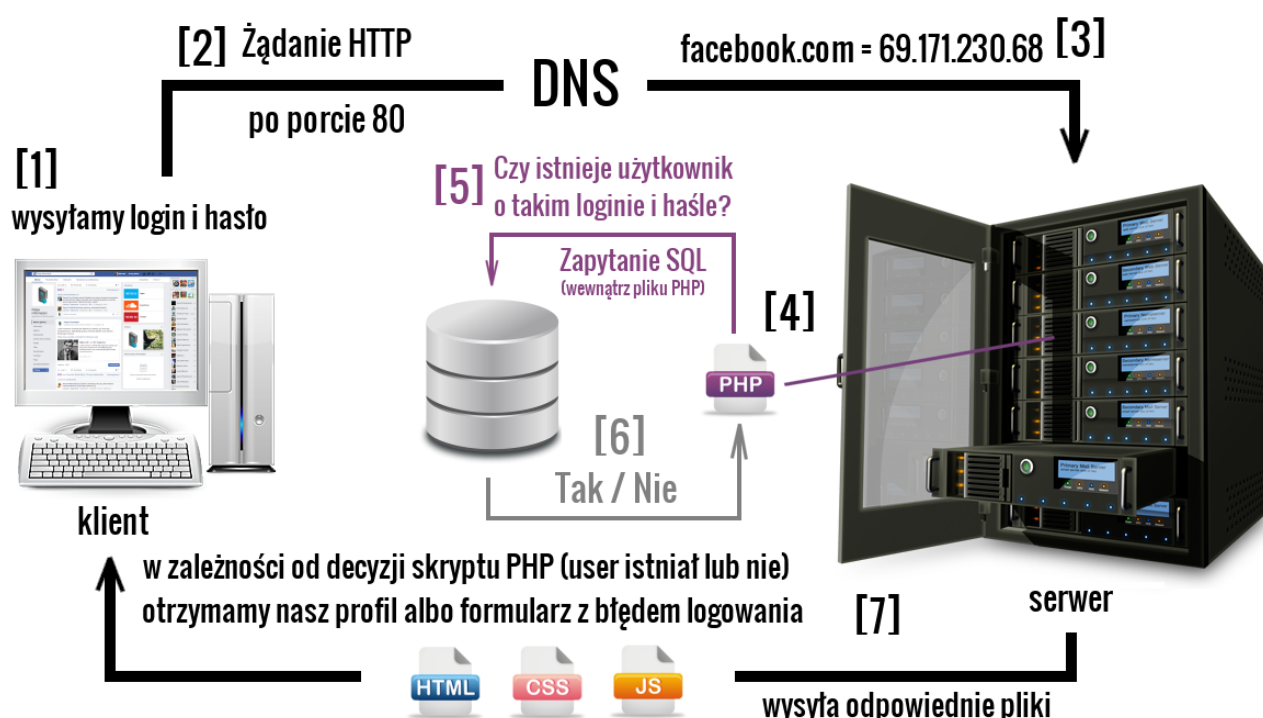
Niemniej jednak nie wszystko przecież można zrobić po stronie klienta. Na przykład fakt oddania łapki w górę pod filmem powinien jednak zostać zapisany w bazie danych na serwerze. Podobnie subskrypcja kanału – serwer musi wiedzieć, że taki fakt miał miejsce. Podobnie kiedy logujemy się w jakimś serwisie, niech będzie Facebook – ten login i hasło trzeba przecież wysłać na serwer, aby tam w serwerowej bazie danych potwierdzić swoją tożsamość. Dopiero po tej weryfikacji można wczytać listę znajomych tego użytkownika, otrzymane wiadomości, powiadomienia oraz wpisy na tablicy. I tu dochodzimy do potrzeby istnienia drugiej strony komunikacji – strony już nie klienckiej, a serwerowej. I tu na scenę wkracza ważny zawodnik – nazywa się: Back-end.

Back-end

Tym mianem określa się technologie webowe, których kody źródłowe wykonywane są przez procesor serwera (czyli w praktyce przez interpreter danego języka zainstalowany na jego dysku twardym). Jako że skrypty te wykonywane są zdalnie, to rzecz jasna do tych kodów nie może zajrzeć każdy internauta – są jak to mówimy: utajnione (musimy posiadać dostęp do dysku twardego serwera, na przykład dzięki usłudze FTP). Języki back-endowe, takie jak PHP zajmują się mechaniką działania serwisów internetowych i dlatego skrypty te stanowią słodką tajemnicę twórców witryn. Do technologii back-endowych zaliczymy przede wszystkim: PHP i SQL.

Prześledźmy działanie back-endu na przykładzie logowania do Facebooka.

Proces logowania przedstawiono na rysunku poniżej:



Rozpocznijmy od prostej realizacji: surfowanie w sieci nie polega tylko na oglądaniu statycznych stron wizytówek. Czasem zechcemy kupić książkę korzystając z koszyka w wirtualnej księgarni, a czasem postanowimy zalogować do serwisu społecznościowego czy skrzynki pocztowej na własne, spersonalizowane konto. Nasz lokalny komputer nie wystarczy do takich działań - to przecież serwer musi nas wpuścić do swojej bazy danych. I dlatego istnieje back-end.

PHP, SQL, MySQL

Pierwszy język back-endowy, z którym będziesz mieć styczność to PHP. Podobnie jak JS, jest to pełnoprawny, skryptowy język programowania, no tyle że wykonywany przez procesor serwera, a nie naszej lokalnej maszyny. PHP obsługuje na serwerze mechanikę (logikę) algorytmu logowania, koszyk w sklepie internetowym, rozbudowę armii w grze przeglądarkowej etc.

Pliki PHP znajdują się tylko na serwerze, w ogóle nie trafią na nasz komputer lokalny. W przypadku logowania wygląda to tak: w naszej przeglądarce uzupełniamy login i hasło i klikamy na przycisk „Zaloguj”. W tym momencie następuje komunikacja z serwerem, z użyciem protokołu HTTP i pośredniczącego serwera DNS – tak samo jak poprzednim razem.

Jednak teraz nasz komputer lokalny nie prosi już tylko o przesłanie mu strony głównej – teraz podaliśmy swój login i hasło serwerowi, oczekując że ten wpuści nas na nasze spersonalizowane konto na Facebooku. Na serwerze znajduje się dokument z rozszerzeniem .php, który zostaje wywołany do pracy. Nazwa tego pliku była przypisana do kliknięcia przycisku „Zaloguj” w formularzu logowania. To stąd serwer wiedział, który konkretny skrypt PHP zawołać w tym przypadku do pracy.

I teraz – skrypt logowania odczytuje przesłane przez nas dane dostępowe (oczywiście skrypt ten jest wykonywany przez procesor serwera), po czym łączy się z bazą danych wszystkich użytkowników sprawdzając, czy znajduje się w niej użytkownik o takim loginie i hasle.

Bazę danych obsługuje dla nas technologia MySQL. Jest to tzw. system zarządzania bazą danych DBMS (ang. Database Management System). Natomiast SQL (ang. Structured Query Language) to język, który służy do pisania zapytań do tej bazy (zapytania również umieszczone są wewnątrz tego samego skryptu z rozszerzeniem .php).

Reasumując: PHP łączy się z bazą MySQL, i kieruje do niej – w języku SQL – zapytanie: „Czy w bazie znajduje się użytkownik o takim loginie i hasle?”. MySQL sprawdza to i odpowiada „Tak!” lub „Nie” (zależnie czy podaliśmy poprawny login i hasło). Bazy danych to potężne i przede wszystkim szybkie silniki składujące olbrzymie ilości danych – nawet gdybyśmy mieli w bazie setki tysięcy zarejestrowanych użytkowników, to i tak MySQL jest w stanie odpowiedzieć na takie zapytanie SQL w ciągu tysięcznych części sekundy!

W zależności od udzielonej przez MySQL odpowiedzi – PHP musi podjąć decyzję. Jeżeli stało się tak, że dane były poprawne i odpowiedź brzmi „Tak” to do klienta zostaje odesłany spersonalizowany profil – panel boczny z avatarą, tablica postów, chat ze znajomymi. Ale uwaga - w jakich językach przychodzi ta odpowiedź? W HTML, CSS i JavaScript! PHP zawsze generuje odpowiedź w językach front-endowych, gdyż ani jedna linia skryptu logowania nie może trafić do przeglądarki klienta.

Po pierwsze byłby to wyciek naszego tajnego algorytmu logowania, a po drugie przeglądarka potraktowałaby kod PHP jak zwykły tekst! Przeglądarka nie interpretuje w ogóle kodu źródłowego PHP czy SQL, bo nie musi. Przeglądarka ma renderować HTML, CSS i JS! Liniami PHP zajmuje się program – interpreter tego języka, zainstalowany na serwerze, co teraz już rozumiemy w pełni.

A teraz pora przyjrzeć się bogactwu różnorodności technologii webowych.

Drzewo technologii webowych

W drzewie technologii ilustrującym możliwe ścieżki rozwoju programisty webowego (które przedstawiono poniżej) obowiązują dwie proste zasady.

Zasada pierwsza: jeśli jakaś technologia znajduje się wyżej od innej, to oznacza to, iż w mojej opinii lepiej jest najpierw opanować to, co umieszczone jest niżej w drzewie. Wyjątkiem jest obszar korzeni drzewa, czyli umiejętności fundamentalne w IT. Tutaj nie obowiązuje zasada że im coś jest wyżej tym później należy to zrealizować, gdyż wszystkie fundamenty są równorzędne i tak samo ważne.

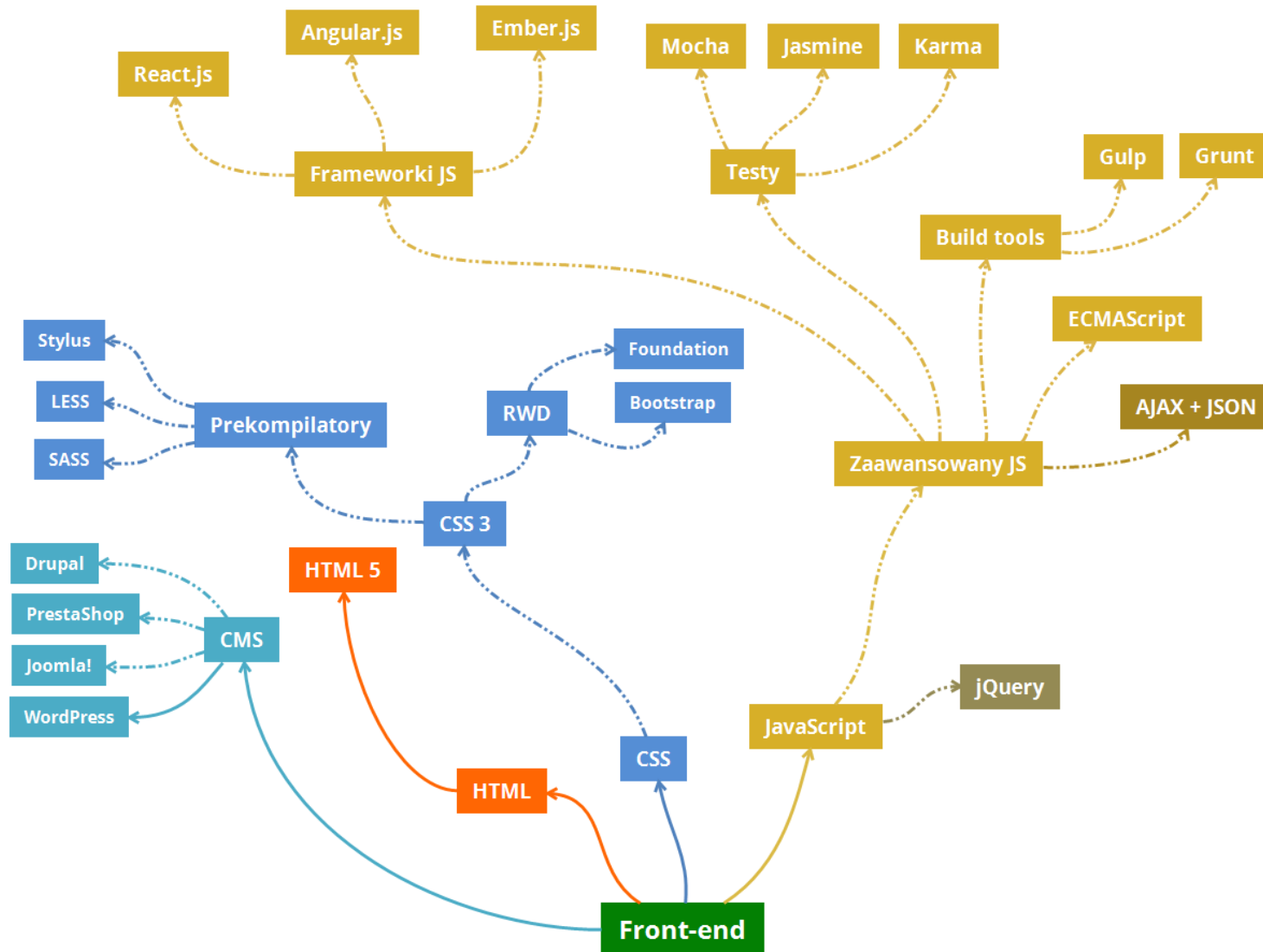
Zasada druga: linie przerywane oznaczają technologie webowe, których dobra znajomość nie jest koniecznie potrzebna na egzaminie zawodowym E.14. Również tutaj wyjątek stanowią umiejętności fundamentalne – nie zostały oznaczone liniami przerywanymi, gdyż dotyczą pracy zawodowej, a nie szkolnego egzaminu.

Na początku ilość i różnorodność technologii webowych na pewno potrafi przytłoczyć. Nie przedstawiłem jednak tak dokładnego podziału webowych technologii, aby Cię zniechęcić - daleki jestem od kładzenia ludziom na barki niemożliwych do uniesienia ciężarów. W nauczaniu (i w osobistej pracy również) stosuje bardzo prostą filozofię, którą można podsumować krótko: nie porównuj siebie do innych – w swojej nauce staraj się być po każdej sesji odrobinę lepszy niż byłeś wczoraj. Rób to wystarczająco długo i cierpliwie, a przyjdą rezultaty.

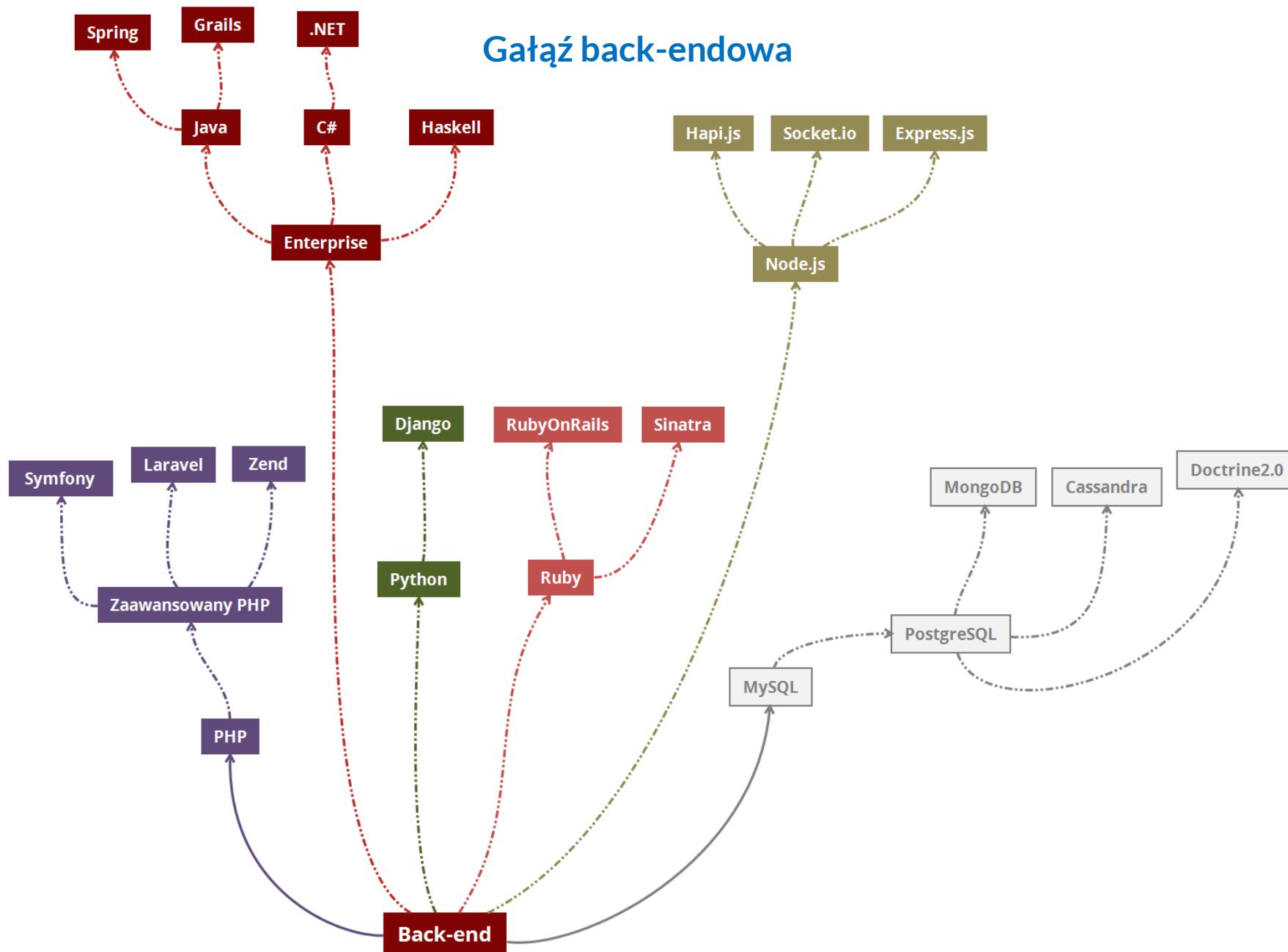
Co do wyboru własnej drogi w drzewie technologii webowych to nikt nie może Ci tego "dać" - sam będziesz wiedzieć, że "to lubię robić". Graficy i osoby o dużym zmyśle estetycznym odnajdą się najszybciej w gałęzi front-endowej, projektując i przycinając layouty stron internetowych, zaś osoby zafascynowane tworzeniem mechaniki (logiki) działania aplikacji internetowych najszybciej odnajdą się w JavaScript i PHP. Zachęcam jednak z czasem eksplorować kolejne języki i frameworki realizowane tak po stronie klienta jak i serwera. Testuj, próbuj, baw się.

Ważne jest zdrowe podejście i świadomość, że nie ma żadnej tajemniczej recepty na sukces. Sukces to progresywna (czyli regularna, nieustająca) realizacja jakiejś wartościowej idei czy pasji. Tak jak powiedział Steve Jobs – nie da się połączyć najważniejszych "kropek" (punktów zwrotnych) w naszym życiorysie zawodowym patrząc w przód, można je połączyć tylko patrząc wstecz. Nigdy nie będzie tak, że będziesz mieć idealny plan i że wszystko będziesz wiedzieć. Trzeba pracować i szlifować swoje umiejętności, a wówczas życie zatroszczy się o naszą przyszłość. Pracuj cierpliwie, pracuj pomimo trudności i własnych ograniczeń. Powodzenia!

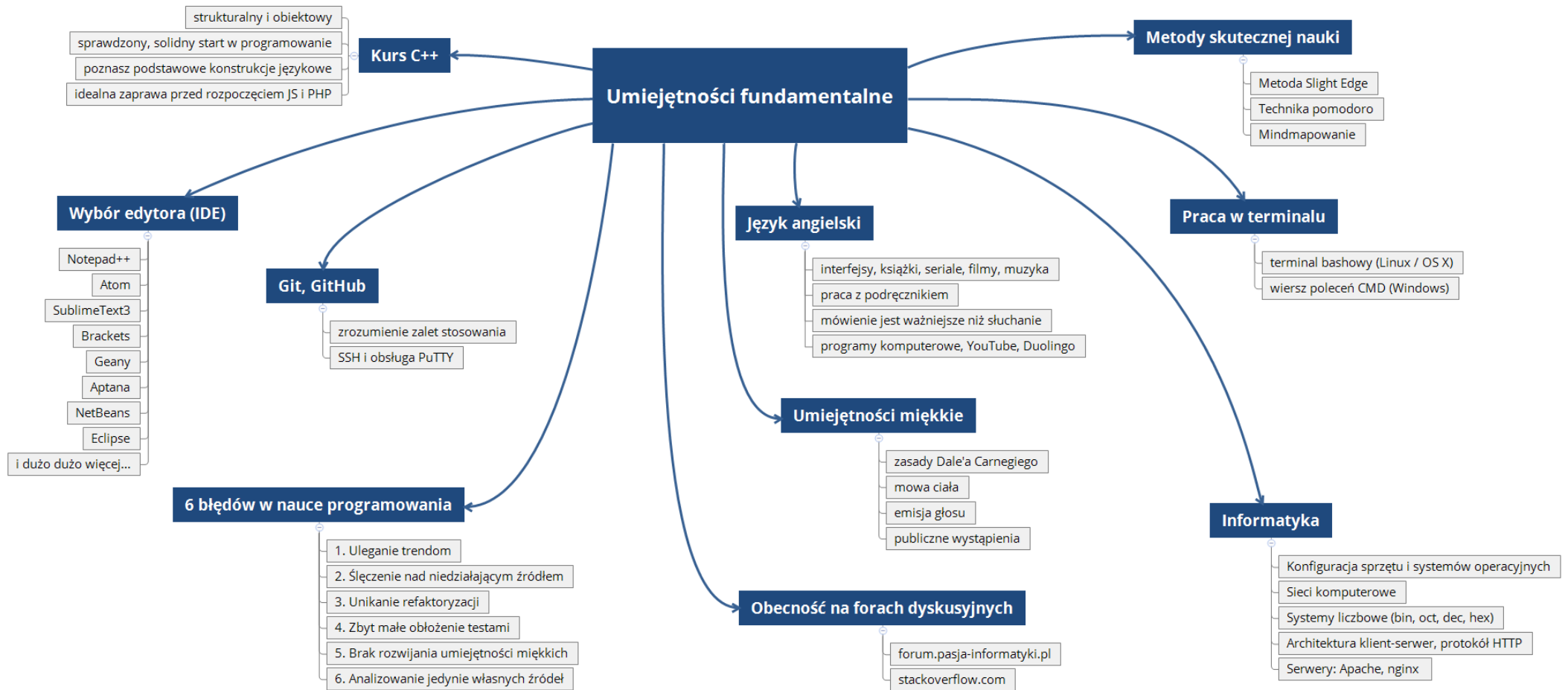
Gałąź front-endowa



Gałąź back-endowa



Umiejętności fundamentalne



Przydatne linki

Poniżej znajdziesz zestaw linków, które przedstawiono także w formie adnotacji w odcinku zerowym serii: <https://www.youtube.com/watch?v=Ugl3ZS62rvs>

Kurs C++

<https://www.youtube.com/playlist?list=PLOYHgt8dlldox0Y5wzs7CFpmBzb40PaDo>

Kurs Obiektowy C++

https://www.youtube.com/playlist?list=PLOYHgt8dlldozvOVheSRb_qPVU-4ZJA7uB

Podcast: 6 najczęstszych błędów podczas nauki programowania

<https://www.youtube.com/watch?v=nXwDEL-ivis>

Arlena Witt

<https://www.youtube.com/user/jezykalnia>

Duolingo

<https://pl.duolingo.com/>

Podcast: Sukces zawodowy w branży IT

<https://www.youtube.com/watch?v=RYb7Qm-qDjM>

Jak opublikować stronę w sieci? Wybór hostingu

<https://www.youtube.com/watch?v=p0CYjMoXilo>

Forum Pasja informatyki

<http://forum.pasja-informatyki.pl/>

Forum Stack Overflow

<http://stackoverflow.com/>

Podcast: Metody skutecznej nauki

<https://www.youtube.com/watch?v=aNqZu2yl9LM>

Kurs sieci komputerowych

<https://www.youtube.com/playlist?list=PLOYHgt8dlldoz2fyn0gv4fs2t4tayalsh3>

Konwersje liczbowe (system dwójkowy, ósemkowy, dziesiętny, szesnastkowy)

<https://www.youtube.com/watch?v=VUHwfugYFEA>

Dwójkowe liczby ujemne: ZM, ZU1, ZU2. Przepelnienie (overflow)

<https://www.youtube.com/watch?v=ZD0wMdJa-Ns>

Kursy dedykowane poszczególnym technologiom webowym:

HTML

https://www.youtube.com/playlist?list=PLOYHgt8dlldox9Qq3X9iAdSVekS_5Vcp5r

CSS

<https://www.youtube.com/playlist?list=PLOYHgt8dlldow6b2Qm3aTJbKT2BPo5iybv>

JavaScript

<https://www.youtube.com/playlist?list=PLOYHgt8ldoxTUYuHS9ZYNlcJq5R3jBsC>

PHP

<https://www.youtube.com/playlist?list=PLOYHgt8ldox81dbm1JWXQbm2geG1V2uh>

MySQL

<https://www.youtube.com/playlist?list=PLOYHgt8ldoymv-Wzvs8M-OsKFD31VTVZ>