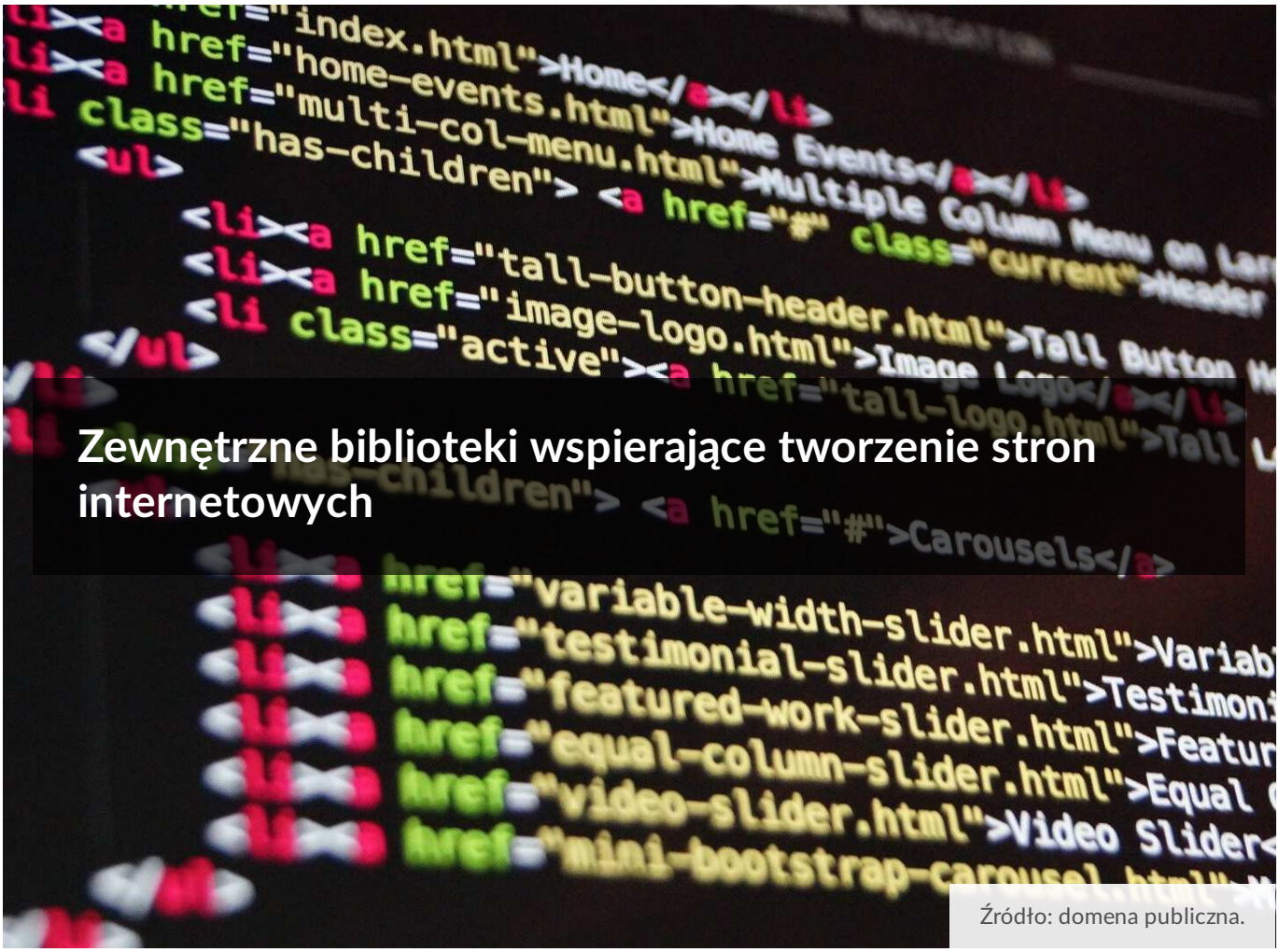
Zewnętrzne biblioteki wspierające tworzenie stron internetowych



Źródło: domena publiczna.

Ludzie już od dawna zajmują się profesjonalnym lub hobbystycznym tworzeniem witryn internetowych. To zupełnie naturalne, że programista wielokrotnie wykonujący daną czynność, z czasem opracuje dla siebie i innych pomocne oprogramowanie, którego użycie znacznie przyspieszy finalizację projektów.

To właśnie ta chęć do automatyzacji własnej pracy, jak również przesłanki zarobkowe (w końcu czas to pieniądz), poskutkowały opracowaniem gotowych systemów i bibliotek, których współcześnie używa wielu programistów webowych na całym świecie. W tym e-materiale przyjrzymy się przeróżnym pomocnym rozwiązaniom, które na stałe zadomowiły się w tak rozległym drzewie webowych technologii.

Twoje cele

- Sklasyfikujesz główne kategorie narzędzi webowych, które wykorzystywane są współcześnie do tworzenia witryn na całym świecie.
- Poznasz i scharakteryzujesz przeznaczenie wielu gotowych bibliotek frameworków i technologii wspomagających programowanie webowe.
- Dowiesz się, jakie są wady i zalety poszczególnych bibliotek.

Przeczytaj

We współczesnym programowaniu webowym wyróżnić możemy kilka podstawowych rodzajów **bibliotek oraz autonomicznych systemów**, których używają adepci sztuki tworzenia witryn na całym świecie. Już podczas nauki programowania webowego, bardzo szybko zauważamy istnienie w tym procesie wielu działań, które **można automatyzować**.

Drzewo dostępnych obecnie technologii webowych oraz gotowych bibliotek do natychmiastowego wykorzystania jest niezwykle rozległe i różnorodne. W obliczu tak wielkiej obfitości i dostępności narzędzi, łatwo jest zagubić się w bogactwie oferowanego nam w sieci oprogramowania. Skrajnie różne, a czasem także sprzeczne bywają również opinie wyrażane przez użytkowników tych systemów.

Dzięki temu e-materiałowi poznasz najbardziej klasyczne biblioteki oraz gotowe systemy tworzenia witryn dostępne na rynku. Wszystkie omówione narzędzia podzielimy na kilka podstawowych kategorii, aby lepiej zrozumieć przeznaczenie każdego systemu oraz biblioteki.

Do podstawowych **kategorii oprogramowania** ułatwiającego, bądź automatyzującego pracę początkującego programisty webowego zaliczamy:

- biblioteki zewnętrzne,
- systemy CMS,
- frameworki front-endowe,
- frameworki back-endowe,
- inne pomocne narzędzia i technologie.

Oczywiście same nazwy tych kategorii niewiele powiedzą nam na tym etapie. Pora więc aby po kolei omówić każdy ze wspomnianych rodzajów oprogramowania, a także przytoczyć nazwy wielu najbardziej rozpoznawalnych reprezentantów każdej z kategorii.

Biblioteki zewnętrzne

Najbardziej klasyczną formą wsparcia procesu tworzenia witryny jest użycie **bibliotek**, czyli **plików zewnętrznych** zawierających całe **zestawy gotowych do użycia funkcji**, realizujących konkretne zadania.

Współcześnie, zdecydowanie największa ilość bibliotek używa języka JavaScript, ale wiele gotowych do wykorzystania funkcjonalności stworzono także w innych językach, np. w CSS.

> **Ciekawostka**

> Kiedy dany kod źródłowy, realizujący wybrane funkcjonalności, przestaje być **zewnętrzną biblioteką**, a staje się **frameworkiem**? Granica bywa płynna, ponieważ ciekawe rozwiązania są często systematycznie rozbudowywane przez społeczności użytkowników do imponujących rozmiarów.
>
> Niemniej jednak, **framework to kompleksowy system**, który jest dokładnie określonym w detalach projektem zbudowania całej witryny (podobnie jak projekt budowy nowego domu).
>
> Biblioteka zaś to w tej metaforze raczej sklep z gotowymi meblami, narzędziami czy farbami – pomoże nam zrealizować **fragment przedsięwzięcia** architektonicznego, użyczając wielu narzędzi, bądź gotowych komponentów.

Poznajmy nazwy kilku popularnych bibliotek:

- `jQuery` – bardzo obszerna biblioteka, która zmienia i usprawnia wiele istotnych aspektów tworzenia witryny. Zastosowanie `jQuery` wpływa na sposób używania selektorów w JavaScript – wprowadzono tzw. funkcję główną `$()`, która upraszcza znacząco wiele zapisów znanych z klasycznego repertuaru JavaScript. Manipulowanie elementami tzw. hierarchii DOM stało się dzięki `jQuery` dużo łatwiejsze! Niegdyś dominująca na rynku, dziś powoli i sukcesywnie wypierana z użycia przez najnowsze frameworki front-endowe, choć niektóre z nich wciąż wykorzystują tę bibliotekę, jak np. `Bootstrap`;
- `Popper.js` – obsługa elementów „wyskakujących" wskutek reakcji na kursor myszki – podpowiedzi, elementy rozwijane, części menu itp.;
- `AOS` – akronim od słów: *Animate on scroll* – biblioteka wykonana w CSS, która zapewnia działanie różnorodnych animacji odtwarzanych podczas przewijania strony;
- `Anime.js` – obsługa złożonych animacji za pośrednictwem JavaScript;
- `FullPage.js` – umożliwia wykonanie witryny, która pozwoli przewijać zawartość płynnie, lecz zawartość całej witryny na bieżącym ekranie;
- `Masonry.js` – zapewnia możliwość użycia w witrynie siatki responsywnych kafelków.

Oczywiście takich pomocnych zestawów funkcji znajdziemy w internecie setki, jeśli nie tysiące. A przecież codziennie powstają nowe. Powyższa lista to jedynie wierzchołek góry lodowej.

Największa zaleta inkludowania (dołączenia) w projekcie biblioteki zewnętrznej to **oszczędność czasu programisty** i dość **niski próg wejścia** – uczymy się jak skorzystać z funkcji, niekoniecznie rozumiejąc w jaki sposób realizuje one swoje zadanie. Ponadto, dzięki zapoznaniu się ze sposobem użycia wybranych metod, zyskujemy możliwość wielokrotnego ich wykorzystania w kolejnych projektach.

Wadą podpinania do kodu witryny zewnętrznych plików bibliotek jest występująca siłą rzeczy **nadmiarowość kodu źródłowego.** Często wykorzystamy jedynie niewielki procent

wszystkich dołączanych w bibliotece funkcji. Witryna zawiera wtedy dużo nieużywanego kodu.

Systemy CMS

Skrót CMS to akronim od ang. *Content Management System*, czyli *System Zarządzania Treścią*. Jest to instalowany na serwerze kompleksowy pakiet, zawierający architekturę całej gotowej witryny, wraz z dodatkowym, wykonanym również jako strona internetowa panelem administracyjnym. Dostęp do takiego zaplecza witryny powinien posiadać tylko **administrator serwisu** oraz (w ograniczonym wymiarze) także jego **redaktorzy**.

Dodawanie nowych treści odbywa się najczęściej z użyciem edytora WYSIWYG. System pozwala na instalację wielu gotowych **wtyczek** (ang. plugin) realizujących konkretne funkcjonalności. Natomiast wygląd strony internetowej można dostosować dzięki istnieniu **szablonów** (ang. theme).

Jest to zatem „gotowiec" w całym znaczeniu tego słowa. Praca z systemem polega w większości na klikaniu oraz obsłudze formularzy panelu administracyjnego. Pisanie własnego kodu oczywiście jest możliwe, lecz jednocześnie z racji istnienia gotowej struktury systemu, nasze możliwości wpływania na sposób jego działania są mocno ograniczone.

Istnieje za to wśród użytkowników scena tworzenia darmowych lub płatnych szablonów (czyli w praktyce zmiany wyglądu strony) do systemów CMS oraz przydatnych wtyczek. Zdolni programiści sprzedają swoje kompatybilne z systemem dodatki, dbając dodatkowo o zapewnienie zgodności kodu w przypadku aktualizacji.

Do **najważniejszych systemów CMS** zaliczamy następujące rozwiązania:

- **WordPress** – najczęściej używany na świecie. Cechuje go łatwość w obsłudze, duża społeczność twórców oraz żywy rynek tysięcy dostępnych wtyczek i szablonów. Od prostego bloga lub strony wizytówki po rozbudowane portale, czy sklepy internetowe – Wordpress oferuje bogatą różnorodność zastosowań;
- **Joomla!** – system dobry do tworzenia portali internetowych, obsługa jego panelu jest jednak nieco trudniejsza i mniej intuicyjna. Z racji mniejszej liczby użytkowników nie oferuje tak olbrzymiej liczby wtyczek i szablonów jak WordPress, stanowi jednak dobrą do niego alternatywę;
- **Drupal** – świetny dla użytkowników średniozaawansowanych, do budowy serwisów o rozbudowanych funkcjonalnościach dla użytkowników. Jest trudniejszy w obsłudze i posiada bardziej rozbudowany panel administracyjny, jednak daje dużo więcej możliwości użycia własnego kodu źródłowego.

Największą wadą tego typu gotowych systemów jest **nadmiarowość kodu źródłowego**, przekładająca się na **zwiększoną wagę plików źródłowych**, a więc mniejszą **szybkość działania w przeglądarce** oraz często **gorsze wyniki w kontekście SEO**. System zapewniający działanie wielu różnorodnych wtyczek oraz mnóstwa funkcji panelu administracyjnego nigdy nie będzie tak lekki i zoptymalizowany jak **dedykowana witryna** – to oczywiste.

Największą zaletą takiego kompleksowego systemu jest możliwość **rozbudowanej konfiguracji** tak **wyglądu,** jak **funkcjonalności** witryny bez znajomości podstawowych technologii i języków webowych: HTML, CSS, JS, PHP i SQL. Nawet osoba będąca kompletnym laikiem w dziedzinie programowania będzie w stanie stworzyć poprawnie działającą witrynę.

Natomiast doświadczony programista, dzięki użyciu gotowych komponentów, zyska możliwość **bardzo szybkiego uruchomienia witryny dla klienta**. Chętnie także wykorzysta olbrzymi popyt na potrzebne użytkownikom wtyczki lub pięknie wyglądające szablony witryn.

Frameworki front-endowe

Rozbudowane systemy tworzenia spójnego wyglądu całej witryny w przeglądarce. Niezależnie od pomysłu na konkretny layout serwisu, framework określa zachowanie wszystkich jego najważniejszych komponentów interfejsu. Mowa tu między innymi o:

- rozwiązaniu kwestii **responsywności całego układu strony;**
- zapewnieniu **intuicyjnej nawigacji** z pomocą menu głównego i pobocznego;
- zorganizowaniu odpowiedniej **konfiguracji typografii**, w tym skalowania rozmiaru nagłówków i akapitów;
- określeniu **spójnego wyglądu** i zachowania podstawowych elementów interfejsów webowych: **kontrolek formularzy**, **tabel**, **obrazów**, **bloku cytatu**, **list numerowanych i nienumerowanych** itd.

Reasumując: framework tego typu to **zestaw bibliotek, plików oraz innych zasobów**, które **realizują według przemyślanej filozofii** kompleksowy i spójny pomysł na sposób zrealizowania interfejsów współczesnej witryny.

Co ważne, frameworki front-endowe nie zajmują się serwerową mechaniką działania witryny, lecz przede wszystkim wyglądem widoków i interfejsów serwisu po stronie klienta. Do najpopularniejszych obecnie tego typu systemów zaliczamy:

- **React** używający głównie języka JavaScript, a stworzony przez twórców portalu Facebook w zamyśle jako pomocnicza biblioteka dla tego serwisu. W 2013 roku kod został udostępniony użytkownikom i wokół zawartych w nim ciekawych technik

(reużywalne komponenty, wirtualna hierarchia DOM) narosła olbrzymia społeczność, która wciąż rozbudowuje i rozszerza zakres jego zastosowania;
- **Angular** stworzony w Google, napisany w języku TypeScript, potężny framework, lecz jednocześnie inny strukturalnie od konkurencyjnych rozwiązań - trudniejszy w opanowaniu dla programisty, ma stosunkowo wysoki próg wejścia;
- **Vue.js** – jego twórcą jest Evan You, były pracownik Google. Łączy on cechy dwóch wcześniej wymienionych systemów: Angular oraz React, a jednocześnie jest lekki i przyjazny użytkownikowi, to znaczy łatwiejszy do nauki nawet dla początkujących programistów;
- **Bootstrap** – początkowo była to biblioteka CSS stworzona przez programistów Twittera, która w ciekawy sposób (z użyciem siatki 12-kolumnowej) rozwiązała problem responsywności całej struktury witryny w kilku ustalonych z góry klasycznych szerokościach ekranu (dzięki użyciu tzw. media queries w CSS). Oddana użytkownikom zdobyła rzeszę sympatyków i rozrosła się do postaci obecnego, bardzo popularnego w sieci frameworka front-endowego.

Kilka pozostałych, wartych uwzględnienia na liście systemów (jak widać ich różnorodność jest olbrzymia i wciąż powstają nowe) to: **Foundation**, **Ember.js**, **Svelte**, **Semantic-UI**, **Backbone.js**, **Materialize**.

Największą **zaletą** użycia frameworków front-endowych jest możliwość **bardzo szybkiego**, **kompleksowego** zapewnienia poprawnego działania wszystkich najważniejszych komponentów serwisu w przeglądarce po stronie klienta. Witryna jest responsywna, ma dobrze zrealizowaną nawigację, a ponadto zadbano o poprawne działanie wszystkich elementów składowych interfejsów, a nawet optymalizację szybkości procesu renderowania widoku w przeglądarce.

Wadą użycia gotowych szkieletów jest stworzenie witryny, która swoim stylem **nie odbiega znacząco od tysięcy innych serwisów** stworzonych w tym samym frameworku. Nawet mało wprawne oko od razu rozpozna kolejną witrynę stworzoną np. w Bootstrapie – cechować ją będzie w wielu elementach interfejsu podobieństwo do wielu widzianych w sieci serwisów.

Frameworki back-endowe

Przeznaczone dla zaawansowanych programistów webowych, kompleksowe systemy realizujące całość serwerowej mechaniki działania serwisu. Wszystkie napisane przez nas funkcjonalności muszą przestrzegać zasad określonych we frameworku – istnienie takiej wspólnej struktury pozytywnie wpływa na jakość i bezpieczeństwo wynikowego kodu.

Tworzenie strony we frameworku back-endowym wymaga wiele wiedzy i doświadczenia. Nie jest to zwykłe pisanie kodu linia po linii, lecz wykorzystanie wielu wysublimowanych technik obiektowych i wzorców projektowych. Skupiamy się nie tyle na działaniu samych

funkcji, co raczej na obsłużeniu całego procesu udzielenia odpowiedzi (ang. response) na żądanie HTTP klienta (ang. request).

Użycie frameworka back-endowego nie definiuje w żaden sposób wyglądu serwisu. Natomiast tego typu kompleksowy system całkowicie przejmuje i nadpisuje rdzenne aspekty działania serwerowej mechaniki witryny – tworzymy tu abstrakcyjne modele kontaktu z bazą danych, używamy opartych na dziedziczeniu metod renderowania poszczególnych widoków podstron oraz kierujemy **routingiem** adresów podstron lub kategorii. Często używamy w tych systemach tak zwanego podejścia MVC.

Podziału frameworków warto dokonać według języka back-endowego, którego dany system używa:

Używany język programowania	Nazwy frameworków back-endowych
PHP	Symfony, Laravel, Zend Framework, CodeIgniter, CakePHP, Yii Framework, FuelPHP, Phalcon
Python	Django, Flask, Tornado, Web2py
Ruby	Ruby on Rails, Sinatra
Node.js	Express.js, Hapi.js, Socket.io, Derby.js
Java	Spring, Grails (Apache Groovy)

Zaletą użycia tak rozbudowanych i kompleksowych systemów jest możliwość **szybkiego implementowania** dobrego jakościowo kodu źródłowego witryny – bezpiecznego, skalowalnego i łatwego w utrzymaniu.

W praktyce, najważniejszą wadą frameworków back-endowych jest dość **wysoki próg wejścia w system dla programisty** – trzeba zrozumieć wiele konceptów, zasad, wzorców architektonicznych, współzależności oraz ról strukturalnych przypisanych do poszczególnych komponentów i bibliotek. A to wszystko pod warunkiem uprzedniego poznania składni danego języka programowania, którego używa framework!

Inne pomocne narzędzia i technologie

Przy okazji omawiania użycia bibliotek zewnętrznych, frameworków oraz systemów CMS warto wspomnieć także o kilku innych rodzajach narzędzi wspomagających:

- **Build tools**, czyli oprogramowanie automatyzujące czynność przygotowania produkcyjnej wersji oprogramowania. Tego typu zdania to na przykład usuwanie znaków białych ze skryptów i arkuszy stylów (co zmniejsza ich rozmiar), optymalizowanie jakości obrazów przy zachowaniu rozsądnego rozmiaru plików albo

przeprowadzenie procesów walidacji i kompilacji kodu. Najpopularniejsze tego typu narzędzia to **Gulp** oraz **Grunt;**
- **Systemy kontroli wersji**, które pomagają zachować porządek wprowadzanych w projekcie zmian, szczególnie gdy tworzony jest przez zespół ludzi. Dzięki istnieniu repozytorium zyskujemy kontrolę nad historią modyfikacji plików źródłowych. Najpopularniejszą na świecie usługą tego typu jest portal **GitHub**, wykorzystujący system kontroli wersji o nazwie **Git**;
- **Systemy zarządzania pakietami**, czyli narzędzia automatycznie pobierające, instalujące i konfigurujące pakiety potrzebnego w danym projekcie oprogramowania. Dbają one także o aktualizacje oraz zapewniają możliwość usuwania pakietów. Kilka często użytkowanych tego typu narzędzi w kontekście programowania webowego to: `npm`, `bower`, `composer`.

Po zakończeniu lektury tej części e-materiału, zachęcamy do zapoznania się z graficzną reprezentacją omówionych systemów i narzędzi, obrazującą praktyczny podział według ich zastosowania.

Słownik

edytor WYSIWYG

ang. *What you see is what you get* – w kontekście webowym określenie edytora, który umożliwia nie tyle pracę z kodem źródłowym HTML, co z wyrenderowanym widokiem witryny; w założeniach, gotowy widok podstrony w przeglądarce klienta, ma być identyczny z prezentowanym w edytorze podglądem

hierarchia DOM

ang. *Document Object Model* – określony przez organizację W3C standardowy model struktury całego dokumentu HTML, wykorzystywany przez współczesne przeglądarki internetowe; model ten jest obiektowy i niezależny od platformy sprzętowej czy używanego w projekcie języka programowania

SEO

akronim od ang. *Search Engine Optimization* – ogół działań, których dokonują programiści webowi w celu poprawy widoczności witryny w wynikach wyszukiwania - w szczególności w najbardziej popularnej wyszukiwarce Google

layout

inaczej szablon serwisu, czyli układ graficzny witryny internetowej; rozmieszczenie nawigacji, treści właściwej i pobocznej, sposób przewijania zawartości, odgrywane animacje, rozmieszczenie elementów interaktywnych, typografię, kolorystykę i zachowanie na urządzeniu mobilnym

routing adresów

konfiguracja zachowania witryny internetowej w zależności od części składowych adresu URL w żądaniu HTTP; na podstawie struktury adresu kontroler podejmuje decyzję, który widok witryny wyrenderować w odpowiedzi na żądanie (np. konkretny wpis blogowy, stronę główną, wybraną kategorię wpisów itd.)

MVC

ang. *Model View Controller* – wzorzec projektowy wykorzystywany w wielu frameworkach; zakłada podział struktury na trzy główne, abstrakcyjne segmenty: model (reprezentacja logiczna rzeczywistości), widoki (interfejsy, warstwa prezentacji) i kontrolery (sterowanie zachowaniem aplikacji)

reużywalność komponentów

cecha, dzięki której komponenty mogą być bardzo łatwo wykorzystane w różnych miejscach, w różnych kontekstach

Mapa myśli

> **Polecenie 1**
>
> Zapoznaj się z mapą myśli przedstawiającą rozwiązania dotyczące bibliotek, frameworków i innych narzędzi omówionych w materiale, a następnie uzupełnij ją o brakujące nazwy.

Uzupełnij mapę myśli.

- Systemy wspierające
  - Biblioteki zewnętrzne
    - jQuery
    -
    -
    -
    - FullPage.js
    - Masonry.js
  - Systemy CMS
    -
    - Joomla!
    - Drupal
  - Frameworki front-endowe
    -
    -
    - Vue.js
    - Bootstrap
    - Foundation
    -
    -
    - Semantic-UI
    - Backbone.js

-
- Frameworki back-endowe
  - PHP
    -
    -
    - Zend Framework
    - CodeIgniter
    - CakePHP
    -
    -
    - Phalcon
  - Python
    -
    -
    - Tornado
    - Web2py
  - Ruby
    - Ruby on Rails
    - Sinatra
  - Node.js
    -
    -
    -
    - Derby.js
  - Java
    - Spring
    - Grails (Apache Groovy)
- Pomocne narzędzia
  - Build tools
    - Gulp

    - Gulp
    - Grunt
- Kontrola wersji
    - Git
    - Github
- Zarządzanie pakietami
    - npm
    - bower
    - composer

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Przyporządkuj wymienione nazwy oprogramowania do pasującej kategorii.

Symfony	framework front-endowy
Vue.js	biblioteka zewnętrzna
WordPress	framework back-endowy
jQuery	menedżer pakietów oprogramowania
Composer	system zarządzania treścią

Ćwiczenie 2

Wstaw odpowiednie akronimy.

[ ] – edytor widoku graficznego gotowej, wyrenderowanej podstrony
[ ] – wzorzec projektowy, stosowany w strukturze frameworków
[ ] – system zarządzania treścią witryny z panelem administracyjnym

MVC SEO WYSIWYG CMS

Ćwiczenie 3

Przypisz nazwy wybranych frameworków do odpowiedniej grupy w zależności od tego, który z języków wykorzystywany jest w danym systemie.

PHP

Python

Node.js

Java

Laravel | Yii Framework | Django | Web2py | Hapi.js | Socket.io | Spring | Symfony | Flask

Ćwiczenie 4

Połącz w pary nazwy frameworków front-endowych z twórcami ich pierwotnych wersji.

React	od twórców serwisu Twitter
Angular	stworzony na potrzeby Facebooka
Bootstrap	napisany przez inżynierów Google

Ćwiczenie 5

Przypisz wadę zastosowania systemu do właściwej nazwy oprogramowania.

[ ] – wysoki próg wejścia, system trudny w analizie i zrozumieniu dla początkującego programisty.

[ ] – ograniczona możliwość wykorzystania własnego kodu źródłowego do wpływu na zachowanie systemu.

[ ] – powtarzalność wyglądu stworzonej witryny względem innych witryn opartych na tym samym systemie.

Bootstrap | Symfony | WordPress

Ćwiczenie 6

Połącz w pary nazwę zewnętrznej biblioteki z funkcją, którą realizuje w witrynie.

`Popper.js`	obsługa elementów „wyskakujących” – podpowiedzi, rozwijanych części menu itd.
`Masonry.js`	możliwość użycia w witrynie siatki responsywnych kafelków
`AOS`	działanie różnorodnych animacji odtwarzanych przy przewijaniu strony
`FullPage.js`	przewijanie strony o całe ekrany, zamiast stopniowo

Ćwiczenie 7

Wskaż tę odpowiedź, która zawiera nazwy tylko frameworków back-endowych.

- ◯ Spring, Drupal, CodeIgniter
- ◯ Ruby on Rails, Express.js, jQuery
- ◯ Laravel, Django, Socket.io
- ◯ Yii Framework, Bootstrap, Tornado

Ćwiczenie 8

Przypisz nazwy wybranych frameworków do odpowiedniej grupy w zależności od ich zastosowania.

Build Tools – przygotowanie wersji produkcyjnej kodu źródłowego

Menedżer pakietów – instalacja, konfiguracja, aktualizacja, usuwanie

Systemy CMS wraz z panelem administracyjnym

bower | Gulp | Joomla! | Drupal | Grunt | npm | WordPress | composer

Dla nauczyciela

Autor: Mirosław Zelent
Przedmiot: Informatyka

Temat: Zewnętrzne biblioteki wspierające tworzenie stron internetowych

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> III. Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi, w tym: znajomość zasad działania urządzeń cyfrowych i sieci komputerowych oraz wykonywania obliczeń i programów.

Treści nauczania – wymagania szczegółowe

> III. Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi.
>
> Zakres podstawowy. Uczeń:
>
> 1) zapoznaje się z możliwościami nowych urządzeń cyfrowych i towarzyszącego im oprogramowania;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Sklasyfikujesz główne kategorie narzędzi webowych, które wykorzystywane są współcześnie do tworzenia witryn na całym świecie.
- Poznasz i scharakteryzujesz przeznaczenie wielu gotowych bibliotek frameworków i technologii wspomagających programowanie webowe.
- Dowiesz się, jakie są wady i zalety poszczególnych bibliotek.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- telefony z dostępem do internetu.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Zewnętrzne biblioteki wspierające tworzenie stron internetowych". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wprowadza uczniów szczegółowo w temat lekcji i jej cele. Może posłużyć się wyświetloną na tablicy zawartością sekcji „Wprowadzenie".
2. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

Faza realizacyjna:

1. Nauczyciel dzieli klasę na grupy. Każda z grup przygotowuje swoją propozycję Mapy myśli.
2. Grupy prezentują przygotowane przez siebie Mapy myśli. Osoby, które akurat nie występują przed klasą, uzupełniają swoje mapy myśli o propozycje koleżanek i kolegów.
3. **Ćwiczenie umiejętności.** Liga zadaniowa - uczniowie wykonują indywidualnie na czas ćwiczenia nr 1-8 z sekcji „Sprawdź się", a następnie omawiają zadania na forum.

Faza podsumowująca:

1. Na koniec zajęć nauczyciel raz jeszcze wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W odniesieniu do ich realizacji dokonuje szczegółowej oceny rozwiązania zastosowanego przez wybranego ucznia.

Praca domowa:

1. Przygotuj słownik pojęć, jakie pojawiły się w e-materiale.

Wskazówki metodyczne:

- Multimedium w sekcji „Mapa myśli" można potraktować jako zadanie domowe dotyczące analizy problemu zawartego w temacie „Zewnętrzne biblioteki wspierające tworzenie stron internetowych".