Tworzenie stron internetowych

https://pixabay.com/pl/illustrations/w-sieci-web-web-developer-1935737/

- Wprowadzenie
- Przeczytaj
- Infografika
- Sprawdź się
- Dla nauczyciela

Źródło: domena publiczna.

Tworzenie stron internetowych to kreatywne i angażujące zajęcie, które rozwija na wielu różnych płaszczyznach – uczy logicznego myślenia, efektywnego analizowania danych oraz ćwiczy nas w cierpliwości podczas znajdowania i poprawiania popełnionych błędów.

Zanim jednak przejdziemy do rzeczywistej pracy z kodem źródłowym, warto przyjrzeć się ścieżce rozwoju programisty stron internetowych, a w szczególności zrozumieć przeznaczenie pięciu podstawowych technologii webowych: HTML, CSS, JavaScript, PHP, SQL.

W tym e-materiale nie zabraknie także omówienia klasycznej webowej dychotomii, czyli wynikającego z architektury klient-serwer podziału technologii na front-end oraz back-end. Dowiemy się także, w jaki sposób dokonać wyboru edytora kodu źródłowego.

Więcej informacji o tworzeniu stron internetowych znajdziesz w e-materiałach:

- Szkielet strony internetowej,
- Język HTML i podstawy CSS.

Twoje cele

- Sklasyfikujesz języki i narzędzia stosowane po stronie klienta oraz na serwerze.
- Zidentyfikujesz przeznaczenie pięciu podstawowych technologii webowych.

- Dobierzesz edytor do zapisywania kodów źródłowych swoich pierwszych stron internetowych.

Przeczytaj

Paradygmat działania klasycznej witryny internetowej nie zmienił się od lat – jest oparty na tzw. **architekturze klient-serwer** i **protokole HTTP**. Często w internecie zauważymy obecność przedrostka `HT` – istnieje w końcu **język tworzenia witryn** o nazwie HTML, a ponadto **tekstowa zawartość stron internetowych** jest przysyłana do przeglądarki klienta **protokołem** HTTP. Co dokładnie oznacza ten skrót `HT`?

Chodzi o `HyperText`, czyli w praktyce **mechanizm działania hiperłączy** (linków) – od lat sześćdziesiątych dwudziestego wieku taki mamy pomysł na działanie stron internetowych. Kliknięcie w hiperłącze przenosi nas do kolejnej podstrony, a czasami także na zupełnie inny serwer. Z tego powodu od razu nasuwa się nam popularne porównanie usługi WWW (ang. World Wide Web) do **pajęczej sieci**.

Aby dobrze zrozumieć sposób tworzenia witryn internetowych, niewątpliwie trzeba najpierw z uwagą przeanalizować i zrozumieć **proces komunikacji** zachodzącej pomiędzy **przeglądarką internetową klienta** serwisu, a **serwerem** wysyłającym w odpowiedzi żądaną witrynę.

Architektura klient-serwer

Kiedy wpisujemy w przeglądarce adres wybranej strony internetowej, np. http://www.facebook.com, to nasz lokalny komputer (klient) zwraca się do przechowującego tę witrynę serwera z tzw. żądaniem `HTTP` (ang. *HTTP request*).

Odpowiedź serwera polega na **dostarczeniu nam plików potrzebnych do wyrenderowania strony** przez naszą przeglądarkę. Plikami odsyłanymi w odpowiedzi (ang. *HTTP response)* są przede wszystkim kody źródłowe zapisane między innymi w językach HTML, CSS oraz JavaScript:

Oczywiście, użytkowanie witryn w sieci **nie polega tylko na oglądaniu statycznych stron** wizytówek. Czasem zechcemy kupić książkę korzystając z koszyka w wirtualnej księgarni lub postanowimy zalogować do serwisu społecznościowego czy skrzynki pocztowej. Korzystamy więc z **własnego, spersonalizowanego konta**. Wówczas lokalny komputer kliencki nie wystarczy do zrealizowania wspomnianych działań – to przecież serwer musi nas wpuścić **do swojej bazy danych**!

I właśnie dlatego istnieje w programowaniu webowym ważny podział technologii na rozwiązania użytkowane **po stronie klienta** – jest to tzw. `front-end` oraz języki i narzędzia **uruchamiane na serwerze**, które określamy mianem `back-end`.

Front-end

Jeżeli w menu kontekstowym przeglądarki wybierzemy opcję *Wyświetl źródło strony*, to ukaże się tekstowy opis zawartości serwisu internetowego, stanowiący drugą „warstwę” przeglądanej przez nas witryny:

```
<!DOCTYPE html>
<html lang="pl" data-wcag-font-size="100">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-sc
    <title>
        Zintegrowana Platforma Edukacyjna
    </title>
    <meta name="description"
```

```
11          content="E-podręczniki to bezpłatne i dostępne dla wszy
12    <link rel="shortcut icon" href="/static/img/favicon.ico">
13    <link rel="icon" href="/static/img/favicon.ico">
14    <script type="text/javascript">(function() { var cpReady = fa
15
```

Znacznik po znaczniku lub jak powiedzą niektórzy: „tag po tagu" kod tekstowy determinuje **strukturę strony**, określa **sposób jej wyświetlania**, definiuje **wszystkie elementy interfejsu**, które zapełnią wirtualne płótno karty przeglądarki.

Widząc takie enigmatyczne teksty, człowiek naturalnie zadaje sobie pytanie: *Co te zapisy oznaczają?* I dlatego programowanie przyciąga ludzi ciekawych świata, u których łączy się **dziecięca fascynacja odkrywania tajemnic** z dorosłym **pragmatyzmem, logiką i kreatywnością**.

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

```
1         chtml: {
2              scale: 1.125,
3              adaptiveCSS: false
4         }
5     };
6 </script>
7 <script type="text/javascript" src="/static/c/mathjax/es5/tex-mml
8 <script type="text/javascript" src="/static/c/main.js?v=1231"></s
9 <link rel="stylesheet" href="/static/c/main.css?v=1231">
10 <link rel="stylesheet" href="/static/c/widget-css.css?v=1231"><li
11
12 <!-- Global site tag (gtag.js) - Google Analytics -->
13 <script async src="https://www.googletagmanager.com/gtag/js?id=UA
```

Pliki źródłowe wykonane w technologiach HTML, CSS i JavaScript są **jawne**, gdyż siłą rzeczy **muszą trafić na nasz komputer lokalny**. To procesor naszej maszyny, a konkretnie lokalna przeglądarka internetowa, zajmuje się wyrenderowaniem wyglądu strony na podstawie przysłanych z serwera plików.

Tak narodziła się nazwa: front-end – tym mianem określa się technologie webowe, do których kodów źródłowych **może zupełnie otwarcie zajrzeć każdy internauta**. Do

technologii front-endowych zaliczymy przede wszystkim: HTML, CSS i w niektórych zastosowaniach także JavaScript.

Back-end

Określenia dotyczy technologii webowych, 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 więc **utajnione**. Aby do nich dotrzeć, musielibyśmy posiadać dostęp do dysku twardego serwera, na przykład dzięki usłudze FTP - taki dostęp posiada właściciel serwisu.

Dzięki **utajnieniu serwerowych kodów źródłowych** nie można więc **skopiować całego serwisu internetowego** wraz z mechaniką jego działania – w końcu to rezultat ciężkiej pracy autorów witryny i to twórcy posiadają prawa autorskie do swojego dzieła.

Do technologii back-endowych zaliczymy przede wszystkim języki: PHP, Python, Ruby, jak również (w niektórych zastosowaniach) JavaScript oraz SQL.

Rozumiejąc już fundamentalny podział języków webowych na te **realizowane po stronie klienta** (front-end) oraz przeciwnie, **po stronie serwera** (back-end) przyjrzyjmy się teraz z osobna każdej z pięciu podstawowych technologii tworzenia witryn.

HTML, czyli zawartość dokumentu

HTML to **język opisowy**, w którym przy pomocy znaczników (inaczej: tagów) określamy **co zawiera dana strona serwisu** – mogą to być elementy takie jak: hiperłącza, obrazki, pola edycyjne, przyciski, tabele, listy numerowane, pojemniki na zawartość, akapity tekstu, nagłówki itd.

> **Ciekawostka**
>
> Znaczna część autorytetów w branży IT oraz autorów podręczników uważa, iż HTML **nie jest językiem programowania** – to prawda, gdyż HTML rzeczywiście jest jedynie **językiem opisowym**. To głównie dlatego, iż nie można go wykorzystać do podjęcia decyzji (instrukcja warunkowa), nie da się także z jego użyciem wykonać pętli, obsłużyć instrukcji wyboru czy zdefiniować własną funkcję.

Wspomniane **znaczniki** (tagi), to specjalne zapisy, które łatwo rozpoznać po obecności **nawiasów ostrych**: < oraz >. Rozważmy przykładowy fragment kodu HTML:

```
1 <h1>Witaj Świecie!</h1>
```

To jest definicja nagłówka (ang. *header*) rozmiaru pierwszego – obecne są tutaj aż dwa znaczniki: **otwierający**: `<h1>` oraz **zamykający**: `</h1>` pomiędzy którymi znalazła się właściwa, tekstowa treść nagłówka. Początki naszej przygody z HTML polegać będą w znacznej mierze na przyswojeniu dużej liczby tagów o różnym zastosowaniu.

Nie przejmuj się jednak, jeżeli zaglądając do kodu źródłowego dowolnej witryny nie rozpoznasz **przeznaczenia poszczególnych znaczników po ich nazwach** – to przyjdzie z czasem, a najwięcej doświadczenia daje zawsze **realizowanie własnych projektów**.

Oprócz zawartości strony, w HTML określamy także **istotne parametry witryny** – na przykład **tytuł** i **opis witryny** w rezultatach wyszukiwania Google, **język i zestaw znaków** właściwy dla danego kraju – w naszym przypadku poprawnie należy zakodować polskie ogonki, np.: ą, ę, ś, ć, ź, ł.

CSS, czyli kaskadowe arkusze stylów

Arkusze stylów (ang. *Cascading Style Sheets*) to specjalne pliki służące do opisania **wyglądu i położenia** elementów witryny, zdefiniowanych uprzednio w HTML. Zawartość arkuszy stylów zapisana jest w języku CSS. Jest to również **język opisowy**, a nie pełnoprawny język programowania.

> **Ciekawostka**
>
> Dlaczego są to tzw. **kaskadowe** arkusze stylów? Otóż kaskada to **rodzaj wodospadu, mającego budowę schodkową**. Jest to wyraźna aluzja do potężnego mechanizmu dziedziczenia stylów przez elementy potomne w hierarchii dokumentu HTML. Mechanizm kaskadowości niejednokrotnie zaoszczędzi nam wiele pracy, gdyż unikniemy dzięki niemu potrzeby redeklaracji wielu zapisów zmieniających wygląd elementów.

W arkuszach stylów odnajdziemy zapisy w postaci:

```
selektor
{
   właściwość: wartość;
}
```

Mając to na uwadze rozważmy przykładowy fragment kodu CSS:

```
h1
{
   color: blue;
```

```
}
```

Zdefiniowany wcześniej w kodzie HTML nagłówek rozmiaru pierwszego został **z użyciem selektora** wybrany z całego dokumentu, po czym zmieniono właściwość (atrybut, cechę) tego obiektu – dokładnie **jego kolor** (ang. color). Tekst nagłówka zostanie zatem zapisany w witrynie niebieską czcionką (ang. *blue*).

W tym momencie nasz dociekliwy umysł zada 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 nagłówków `h1` w witrynie.

Staje się to możliwe, gdy **wszystkie podstrony** serwisu używają **tego samego arkusza stylów**. To potężny i genialny w swojej prostocie mechanizm i naturalna ewolucja od czasów gdy wygląd elementów witryny również określało się w HTML, np. z wielokrotnym używaniem tagów `<font>`:

```
<font color="blue">Niebieski tekst</font>
```

Na szczęście czasy miksowania **zawartości witryny** z warstwą jej **wyglądu** dzięki językowi CSS bezpowrotnie minęły.

Dodatkowe funkcjonalności po stronie klienta

JavaScript 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. Jest to więc przede wszystkim potężny front-endowy silnik logiczny, za pomocą którego możemy tworzyć wyjątkowe od strony designu i interfejsu witryny.

> **Ważne!**
>
> JavaScriptu nie należy mylić z językiem Java, gdyż to zupełnie inna technologia.

Na samym początku przygody z tworzeniem stron internetowych, JavaScript używany jest najczęściej do **ulepszenia interfejsu** strony, wzbogacając ją o **dodatkowe funkcjonalności**, niedostępne w HTML czy CSS. Skrypty wykonane w JS pozwolą nam tworzyć efektowne slidery, animowane galerie zdjęć, wyskakujące panele z nawigacją, interaktywne menu, zegary odświeżające się co sekundę, animacje itd.

JavaScript umożliwia także zmianę wybranego fragmentu kodu HTML czy stylów CSS w dokumencie **bez potrzeby nawiązania ponownej komunikacji z serwerem**, czyli bez

konieczności wysłania żądania HTTP zakończonego odświeżeniem całej zawartości płótna przeglądarki. Na ekranie możemy więc niejako „na żywo” dokonać przeróżnych działań:

- zmienić slajd prezentacji,
- kontynuować odliczanie zegara co sekundę,
- pokazać w galerii zdjęć widok pełnej fotografii po kliknięciu na miniaturkę,
- wysunąć dodatkowy panel z menu nawigacyjnym w animowany sposób.

Jako pełnoprawny język programowania 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 np. upływ zadanej ilości czasu, wczytanie zawartości plików witryny lub (najczęściej) kliknięcie czegoś. Zadaniem programisty jest więc **przygotować kod własnej funkcji** realizującej odpowiednią **obsługę** takiego zdarzenia.

Rozważmy jako przykład serwis *YouTube* – inne działania zostaną wykonane w witrynie, kiedy na przykład:

- klikniemy przycisk pauzy na pasku narzędziowym odtwarzania filmu,
- przyznamy łapkę w górę pod ciekawym filmem,
- prześlemy komentarz o treści: *Pierwszy!*,
- zasubskrybujemy treści wybranego kanału,
- klikniemy miniaturkę nowego teledysku w sekcji polecanych filmów,
- zmienimy rozmiar okna przeglądarki.

Najlepsze witryny w sieci zdecydowanie są **interaktywne i żywo reagują na nasze poczynania**. I dlatego właśnie potrzebujemy potężnego front-endowego języka programowania, który przejmie **obsługę** tego wszystkiego, co użytkownik może w interfejsie witryny np. wskazać, kliknąć, wpisać, rozwinąć, zmienić rozmiar itd.

Ponadto, ważny mechanizm responsywności stron internetowych również jest podszyty kodem źródłowym skryptów JS, realizujących odpowiednie zmiany rozmiaru i rozmieszczenia elementów.

Koniecznie zwróćmy jednak uwagę, że **nie wszystkie działania internauty można obsłużyć jedynie po stronie klienta**. Na przykład fakt oddania łapki w górę pod filmem powinien jednak zostać zapisany w **bazie danych na serwerze**. Podobnie subskrypcja kanału, logowanie na własne konto, wysłanie wiadomości etc. I tu na scenę wkraczają technologie back-endowe, realizowane na serwerze.

> **Ważne!**
>
> JavaScript nie jest językiem tylko front-endowym. Owszem, na samym początku swojej przygody, używamy go najczęściej do polepszenia interaktywności interfejsu witryny w przeglądarce klienta. Z czasem jednak przekonamy się, że to był jedynie wstęp do

> prawdziwej potęgi tego języka – w wielu zastosowaniach JavaScript spełnia obecnie także rolę back-endową. Są to między innymi technologie: `AJAX` oraz `Node.js`.

Mechanika działania na serwerze

Pierwszy język back-endowy, z którym będziesz mieć styczność to PHP. Podobnie jak JavaScript jest to **pełnoprawny, skryptowy język programowania**, jednak wykonywany przez **procesor serwera**, a nie **lokalnej maszyny** klienta.

PHP obsłuży na serwerze mechanikę (logikę) algorytmu logowania, zapewni obsługę koszyka w sklepie internetowym, wysłanie wiadomości prywatnej w serwisie społecznościowym etc.

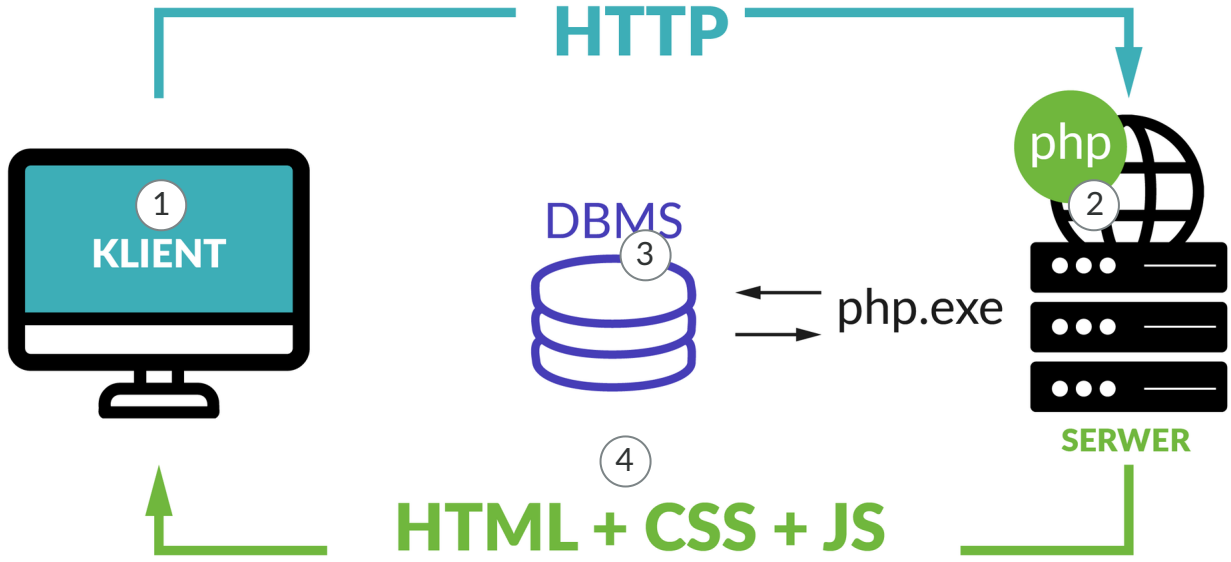
Pliki (skrypty) PHP znajdują się **tylko na serwerze**, w ogóle nie trafią na komputer lokalny.

Rozważmy proces logowania do dowolnego serwisu, który umożliwia posiadanie własnego konta. W naszej przeglądarce **uzupełniamy wówczas login i hasło**, po czym klikamy na przycisk z napisem *Zaloguj*. W tym momencie następuje komunikacja z serwerem z użyciem protokołu HTTP.

Jednak teraz nasz komputer lokalny nie prosi już tylko o przesłanie mu strony głównej – teraz przysłaliśmy serwerowi także swój login i hasło, **oczekując że ten wpuści nas na nasze spersonalizowane konto**.

Na serwerze znajduje się skrypt z rozszerzeniem `.php`, który zostaje **wykonany przez procesor serwera**. Nazwa tego pliku była przypisana do kliknięcia przycisku *Zaloguj* w formularzu logowania. To stąd serwer wiedział, który konkretny skrypt PHP wywołać w tym przypadku do pracy.

Podczas przetwarzania kodu skryptu logowania przez procesor serwera, **odczytane są przesłane przez nas dane dostępowe** i **następuje połączenie z bazą danych**, celem sprawdzenia czy znajduje się w niej użytkownik o takim loginie i haśle. O technologiach obsługujących działanie bazy danych porozmawiamy już w następnej sekcji tego e-materiału.



1

Klient uzupełnia własne dane logowania na stronie, generując tzw. żądanie HTTP.

2

Na dysku serwera znajduje się żądany przez klienta skrypt – kod źródłowy PHP znajdujący się w dokumencie zostaje przetworzony przez program `php.exe` znajdujący się również na dysku twardym serwera.

3

Skrypt PHP pobiera i przetwarza dane logowania i z użyciem języka SQL komunikuje się z bazą danych, która zwraca odpowiedź na pytanie czy istnieje w bazie użytkownik o takich danych dostępowych.

4

Po przetworzeniu całego kodu skryptu, do użytkownika zostaje wysłana odpowiedź – instrukcja warunkowa zapisana w kodzie PHP zwróciła pliki profilu użytkownika, bądź w przypadku nieudanego logowania przekierowała z powrotem do formularza kolejnej próby logowania.

Natomiast w zależności od udzielonej przez bazę danych odpowiedzi, skrypt PHP musi **podjąć decyzję**: jeżeli dane **były poprawne** (istnieje użytkownik o takim loginie i haśle) to

do klienta zostaje odesłany spersonalizowany profil osobisty – **udało nam się zalogować!**

Lecz jeżeli dane logowania **okazały się błędne**, to skrypt PHP przekieruje internautę z powrotem do formularza logowania, **wyświetlając dodatkowo informację, że wprowadzono niepoprawny login lub hasło**.

To jest właśnie przykład mechaniki realizowanej po stronie serwera - to między innymi **podejmowanie decyzji** oraz obsługa **wprowadzania lub wyjmowania informacji z bazy danych**. Technologie back-endowe możemy sobie wyobrazić jako serwerowy silnik obsługujący **zdalne** funkcjonalności aplikacji internetowej.

Oprócz popularnego języka PHP, do realizowania funkcjonalności serwerowych możemy użyć między innymi tych alternatywnych języków programowania: Python, Ruby, Java, C#, Go, Rust, Node.js.

Zaplecze bazodanowe serwisu

Bazy danych stanowią kluczowy element funkcjonowania wielu aplikacji internetowych, gdyż **przechowują one informacje niezbędne do jej działania**. Np. internetowa księgarnia przechowuje w bazie danych konta użytkowników (w tym ich dane osobowe, adresy mailowe, hasła), historię dokonanych przez klientów zamówień, jak również wszystkie możliwe do zakupienia książki.

Ta warstwa działania witryny to tzw. **system zarządzania bazą danych**, w skrócie DBMS (ang. *Database Management System*). Do najpopularniejszych rozwiązań obecnych na rynku należą systemy: MySQL firmy Oracle wraz w darmową wersją MariaDB, PostgreSQL opracowany na uniwersytecie w Berkeley oraz FireBird korporacji Borland.

Natomiast SQL (ang. Structured Query Language) to **język zapytań** (kwerend) wysyłanych do bazy danych, który jest wykorzystywany przez zdecydowaną większość współczesnych systemów DBMS.

Zapytania najczęściej **umieszczamy wewnątrz skryptów back-endowych** (na przykład skryptów PHP, gdyż to one współpracują z bazami danych celem wygenerowania odpowiedzi HTTP na żądanie użytkownika.

Podsumowanie

Gratulacje! Właśnie poznaliśmy i omówiliśmy **pięć podstawowych aspektów** działania witryny internetowej:

1. Zawartość dokumentu (HTML).
2. Wygląd i położenie elementów strony (CSS).
3. Skryptowy język programowania po stronie klienta (JavaScript).

4. Skryptowy język programowania na serwerze (np. `PHP`, `Python`, `Ruby`).
5. System zarządzania bazą danych (np. `MariaDB`, `PostgreSQL`) wraz ze strukturalnym językiem zapytań `SQL` obsługującym komunikację z bazą danych.

Pamiętajmy także, że cała ta synergiczna współpraca wymienionych powyżej technologii oparta jest na fundamencie tzw. **architektury klient-serwer** oraz na działaniu **protokołu HTTP**.

Ostatnim aspektem tworzenia witryn internetowych, który omówimy w tym materiale jest **wybór narzędzia** do **pisania swoich pierwszych kodów źródłowych** we wspomnianych pięciu podstawowych technologiach.

Wybór edytora kodu źródłowego

Zanim będzie można dokonać wyboru odpowiedniego narzędzia do pracy, warto wyjaśnić różnicę pomiędzy edytorem kodu a tzw. środowiskiem IDE.

Edytor kodu źródłowego to program komputerowy umożliwiający nam wygodne pisanie linii kodu: podświetla składnię, numeruje kolejne linie, umożliwia zaawansowaną edycję źródła witryny (wyszukiwanie i zamiana tekstu, system podpowiedzi, konfiguracja kodowania znaków).

Często też edytory pozwalają nam zastosować tzw. tematy (skórki) – otrzymujemy do wyboru wiele różnych **szablonów kolorystycznych**, co ułatwia spersonalizowanie procesu pisania własnego kodu.

Przykłady popularnych edytorów kodu:

- Notepad++,
- Sublime Text 2,
- Atom,
- Microsoft Visual Studio Code,
- KompoZer,
- Brackets.

IDE (ang. *Integrated Development Environment*), czyli tzw. *zintegrowane środowisko programistyczne*. Jest to już nie tylko edytor kodu, ale cały zestaw narzędzi pozwalających między innymi: wygodnie debugować kody źródłowe, testować tworzone skrypty i dbać o porządek w strukturze dyskowej i etapach realizacji dużych projektów.

Ważnym czynnikiem przy wyborze może okazać się także **wysoka cena** takiego zaawansowanego zestawu narzędzi, choć oczywiście istnieją **wersje darmowe do użytku akademickiego**.

Przykłady popularnych zintegrowanych środowisk programistycznych:

- PhpStorm,
- Microsoft Visual Studio,
- Eclipse,
- Aptana Studio,
- NetBeans.

Oczywiście różnica pomiędzy edytorem a środowiskiem IDE **często bywa płynna**.
W internecie spotkamy się także z próbami uczynienia z edytora kodu zintegrowanego środowiska poprzez dołączanie do jego wersji podstawowej dużej liczby wtyczek, stale opracowywanych przez społeczność zgromadzoną wokół danego narzędzia.

Podobnie, jeśli na początku swojej przygody używamy tylko elementarnych opcji środowiska IDE, to w zasadzie **niczym nie różni się to od używania prostego edytora**. Nie jest też tak, iż pisanie w IDE automatycznie uczyni z kogokolwiek lepszego programistę.

Co zatem wybrać? Na poziomie szkoły średniej, a już na pewno w pierwszych kontaktach z tworzeniem stron internetowych, wystarczy **jakikolwiek edytor podświetlający składnię** i posiadający **ciemną, oszczędzający oczy skórkę tła** (oczywiście nic nie stoi na przeszkodzie, aby było to od razu IDE). Reguła jest prosta: dokonaj wyboru na podstawie subiektywnego odczucia **przyjemności pisania kodu** w danym programie.

Nie poświęcaj jednak zbyt wiele uwagi bawieniu się tematami kolorystycznymi oraz dodatkowymi funkcjami oferowanymi przez wtyczki edytorów – **przede wszystkim pracuj z kodem!**

Do **świadomego wyboru** swojego docelowego środowiska naturalnie powrócisz na etapie szlifowania własnych umiejętności, czyli już po opanowaniu podstaw tworzenia witryn.

Słownik

> **strona statyczna**
>
> witryna internetowa pasywnie prezentująca treść, nie udostępniająca form personalizowania doświadczenia dla użytkownika np. poprzez możliwość założenia własnego konta w serwisie; przykładem może być prosta strona-wizytówka osoby lub firmy
> **usługa FTP**
>
> (ang. *File Transfer Protocol*) – protokół transferu plików, jest to wspomagana dedykowaną aplikacją możliwość transferu plików z komputera lokalnego na dysk serwera
> **debugowanie**

(ang. *debugging*) – wspomagany komputerowo proces systematycznego lokalizowania oraz redukowania liczby błędów w oprogramowaniu; program wspomagający i automatyzujący ten proces nazywamy z j. ang. debuggerem

temat, skórka

(ang. *theme*) – plik lub zestaw plików konfiguracyjnych, które zmieniają wygląd interfejsu edytora kodu źródłowego, w tym szczególnie schemat kolorów używanych do podświetlania składni wybranego języka oraz tła całego pola tekstowego w programie

wtyczka

(ang. *plugin*) – moduł z dodatkowymi plikami, które rozszerzają możliwości oryginalnego edytora pobranego i zainstalowanego w jego wersji podstawowej

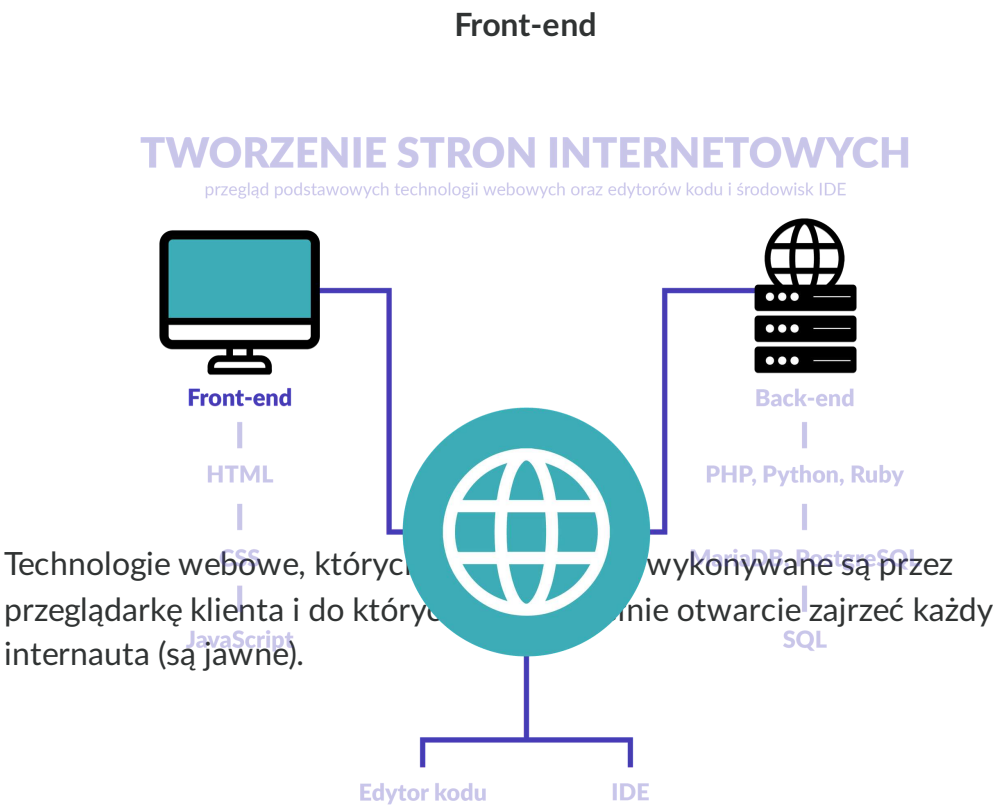
responsywność

wspierany przez języki CSS i JavaScript mechanizm dopasowywania sposobu wyświetlania strony internetowej do aktualnie dostępnego rozmiaru okna przeglądarki; szczególnie istotnie wpływa na poprawę komfortu użytkowania witryny na urządzeniach mobilnych

Infografika

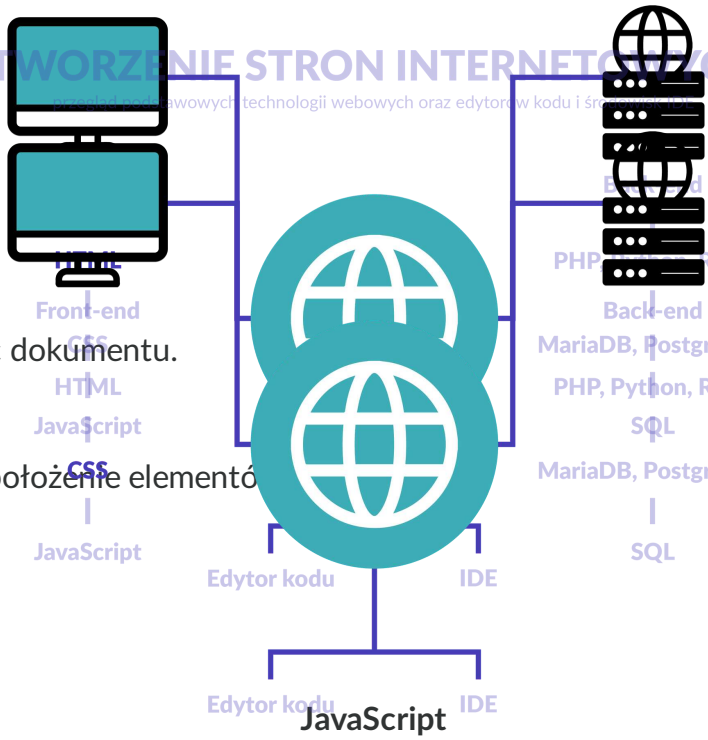
Polecenie 1

Zapoznaj się z infografiką.



Technologie webowe, którycl wykonywane są przez przeglądarkę klienta i do którycl nie otwarcie zajrzeć każdy internauta (są jawne).

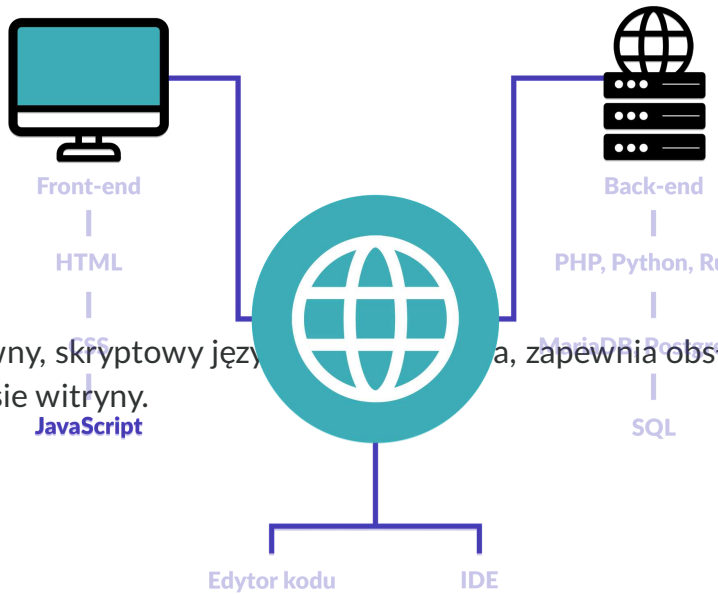
CSS



Zawartość dokumentu.

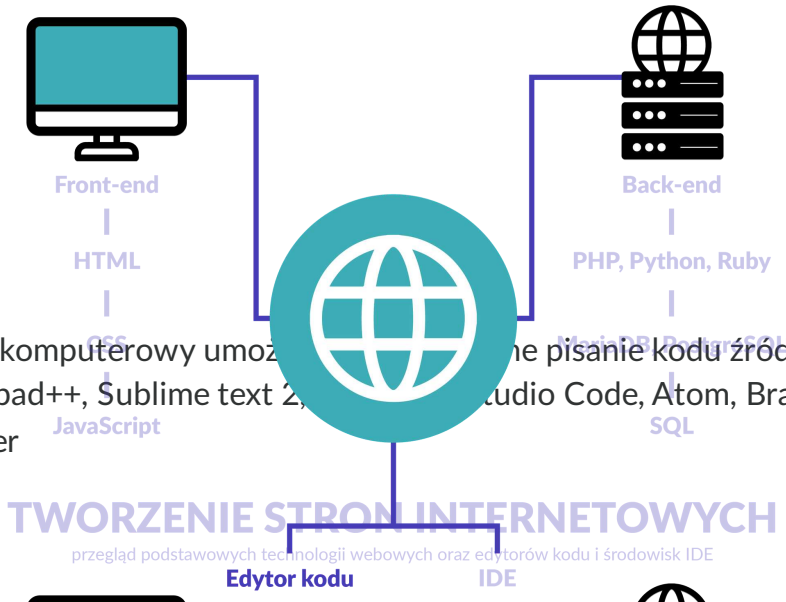
Wygląd i położenie elementó

JavaScript



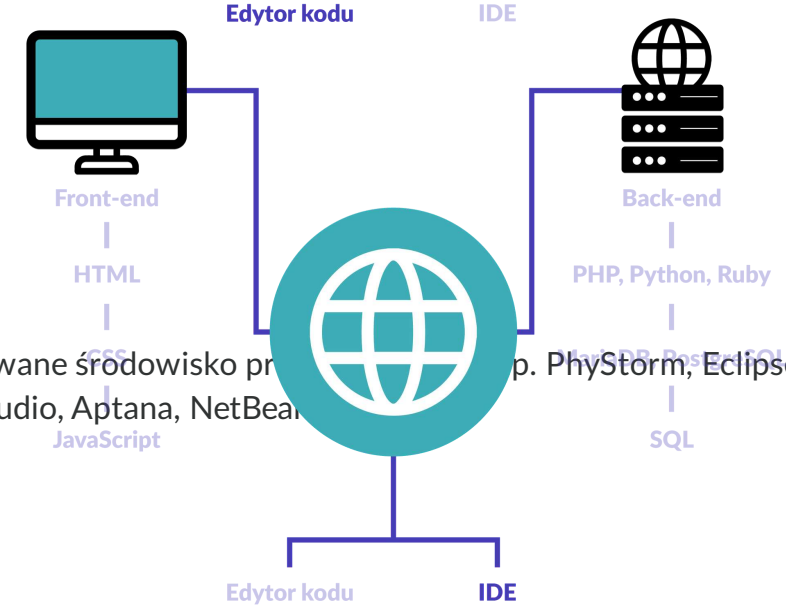
Pełnoprawny, skryptowy języ a, zapewnia obsługę zdarzeń w interfejsie witryny.

Edytor kodu



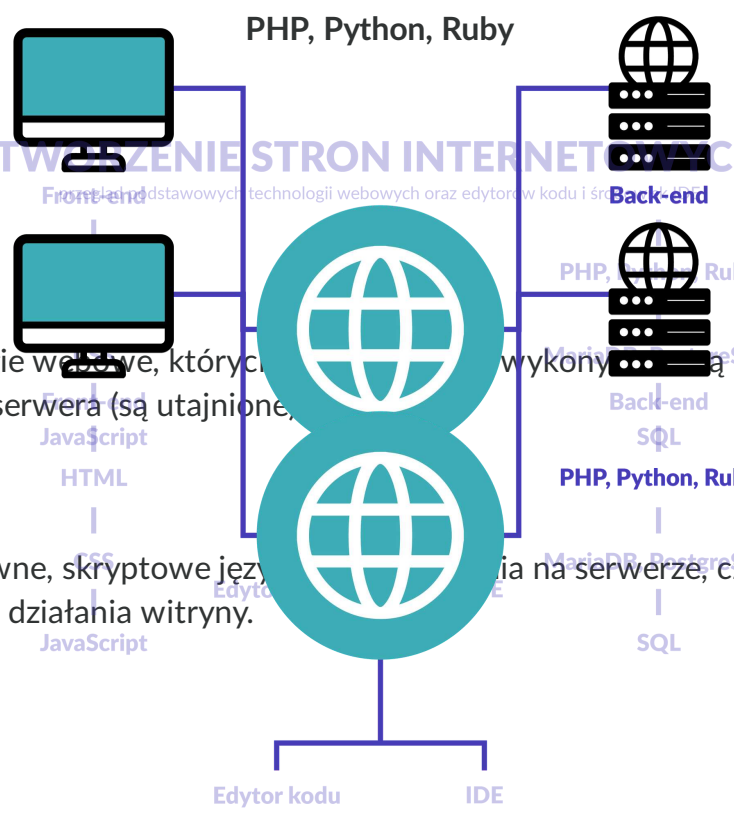
Program komputerowy umoż ne pisanie kodu źródłowego, np. Notepad++, Sublime text 2, udio Code, Atom, Brackets, KompoZer

TWORZENIE STRON INTERNETOWYCH
przegląd podstawowych technologii webowych oraz edytorów kodu i środowisk IDE



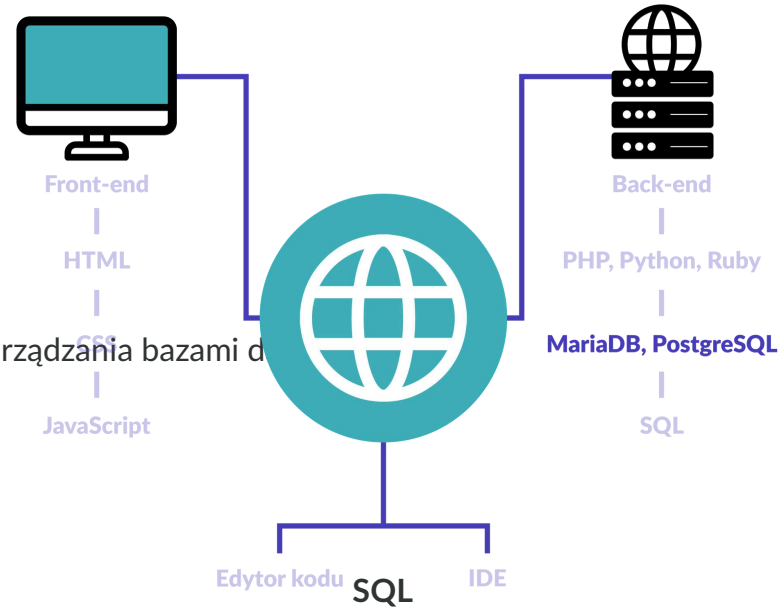
Zintegrowane środowisko pr p. PhyStorm, Eclipse, MS Visual Studio, Aptana, NetBea

Back-end

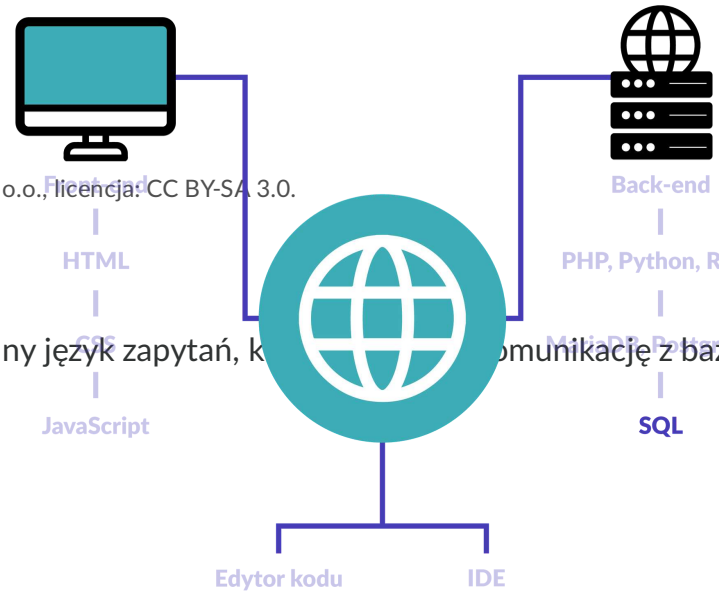


Technologie webowe, którycz wykonywane są przez procesor serwera (są utajnione)

Pełnoprawne, skryptowe języ ia na serwerze, czyli obsługa mechaniki działania witryny.



System zarządzania bazami d



Strukturalny język zapytań, k omunikację z bazami danych.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Połącz w pary poniższe języki z właściwym dla nich opisem:

SQL	język opisowy, określający jak wyglądają elementy znajdujące się na stronie
CSS	język zapytań kierowanych do bazy danych współpracującej z witryną
PHP	pełnoprawny, skryptowy język programowania wykonywany u klienta
HTML	język opisowy, określający jakie elementy znajdują się na stronie
JavaScript	pełnoprawny, skryptowy język programowania wykonywany na serwerze

Ćwiczenie 2

Wskaż poprawną odpowiedź. Aby opublikować witrynę w internecie, należy jej pliki źródłowe skopiować na serwer za pomocą protokołu/usługi:

- ◯ AJAX
- ◯ HTCP
- ◯ FCS
- ◯ FTP

Ćwiczenie 3

Przypisz podane poniżej nazwy oprogramowania do właściwej grupy ich zastosowania:

Zintegrowane środowisko programistyczne

Edytor kodu źródłowego

System zarządzania bazą danych

Microsoft Visual Studio Code

PostgreSQL

Microsoft Visual Studio

MariaDB

Notepad++

PhpStorm

MySQL

Sublime Text 2

Ćwiczenie 4

Wskaż wszystkie zdania PRAWDZIWE:

- [ ] JavaScript nie jest językiem tylko front-endowym.
- [ ] SQL to język komunikacji z arkuszami stylów.
- [ ] PHP to język skryptowy interpretowany po stronie klienta.
- [ ] HTML to pełnoprawny, skryptowy język programowania.
- [ ] CSS to język tylko opisowy.

Ćwiczenie 5

Wstaw do definicji odpowiednie, brakujące elementy:

Do skryptów front-endowych [ ] zajrzeć każdy internauta, są [ ].
Kody źródłowe tych skryptów wykonywane są przez procesor [ ].

Do skryptów back-endowych [ ] zajrzeć każdy internauta, są [ ].
Kody źródłowe tych skryptów wykonywane są przez procesor [ ].

klienta | może | utajnione | jawne | serwera | nie może

Ćwiczenie 6

Wskaż poprawną odpowiedź. Co to jest DBMS?

- ◯ kaskadowy arkusz stylów do opisu wyglądu witryny
- ◯ język zapytań kierowanych do bazy danych
- ◯ system zarządzania bazą danych
- ◯ skryptowy język programowania po stronie serwera

Ćwiczenie 7

Wstaw podane poniżej języki do właściwej grupy, zgodnie z ich przeznaczeniem:

Technologie front-endowe

Technologie back-endowe

Node.js | SQL | Ruby | PHP | HTML | Python | CSS

Ćwiczenie 8

Wskaż listę zawierającą jedynie technologie back-endowe:

- ◯ Node.js, AJAX, MySQL
- ◯ HTML, CSS, JavaScript
- ◯ Python, PHP, CSS
- ◯ Ruby, HTML, PostgreSQL

Dla nauczyciela

Autor: Mirosław Zelent
Przedmiot: Informatyka

Temat: Tworzenie stron internetowych

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres podstawowy

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> Zakres podstawowy. Uczeń:
>
> 3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:
>
> f) tworzy stronę internetową zgodnie ze standardami, wzbogaconą tabelami, listami, elementami dynamicznymi, posługuje się arkuszem stylów, korzysta z oprogramowania i serwisów przeznaczonych do tworzenia stron; potrafi opublikować własną stronę w internecie;

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 języki i narzędzia stosowane po stronie klienta oraz na serwerze.
- Zidentyfikujesz przeznaczenie pięciu podstawowych technologii webowych.

- Dobierzesz edytor do zapisywania kodów źródłowych swoich pierwszych stron internetowych.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

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

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.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Tworzenie stron internetowych". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel inicjuje rozmowę wprowadzającą w temat lekcji. Przedstawia cele zajęć oraz kryteria sukcesu.

Faza realizacyjna:

1. **Praca z tekstem.** Jeżeli przygotowanie uczniów do lekcji jest niewystarczające, nauczyciel prosi o indywidualne zapoznanie się z treścią zawartą w sekcji „Przeczytaj". Każdy uczestnik zajęć podczas cichego czytania wynotowuje najważniejsze kwestie poruszane w tekście.

2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Infografika". Uczniowie wspólnie zapoznają się z treścią zawartego w niej multimedium.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenia nr 1-8. Po wykonaniu każdego z nich następuje omówienie rozwiązania przez nauczyciela.

Faza podsumowująca:

1. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.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. Uczniowie dokonują walidacji przykładowego kodu HTML za pomocą narzędzi on-line.

Wskazówki metodyczne:

- Multimedium w sekcji „Infografika" można potraktować jako zadanie domowe dotyczące analizy problemu zawartego w temacie „Tworzenie stron internetowych".