

Temat 3 .

dr inż. Michał Malinowski

<https://www.drmalinowski.edu.pl>

Komunikacja z serwerem WWW

TECHNIKI INTERNETOWE



Zagadnienia:

- GET, POST
- jQuery
- AJAX
- Synchroniczność, Asynchroniczność
- Cookie
- MIME
- FTP
- Uwierzytelnienie



Test Turinga - test na inteligencję maszyn



Rozmówca A

Rozmówca B



Człowiek

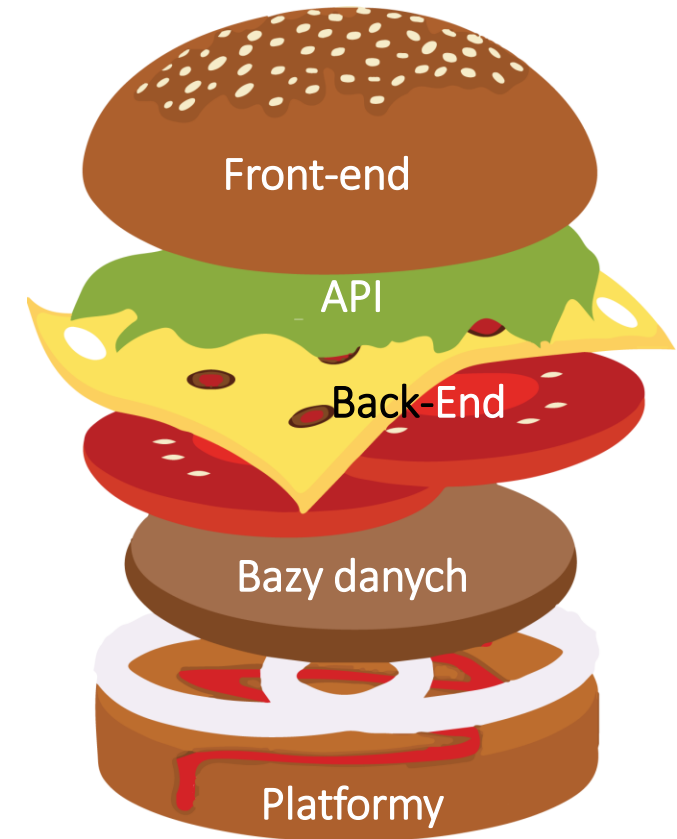
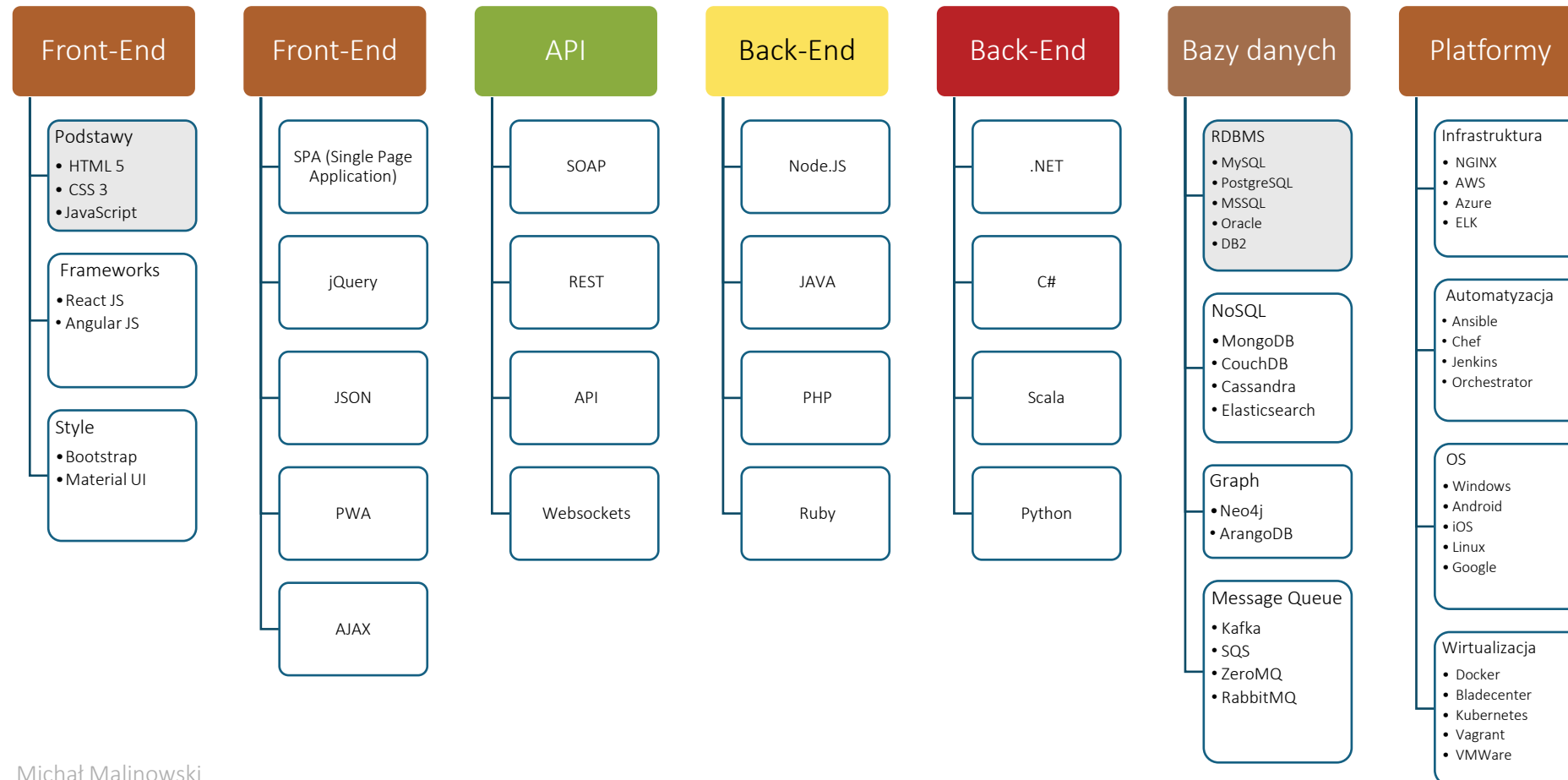


Sztuczna Inteligencja

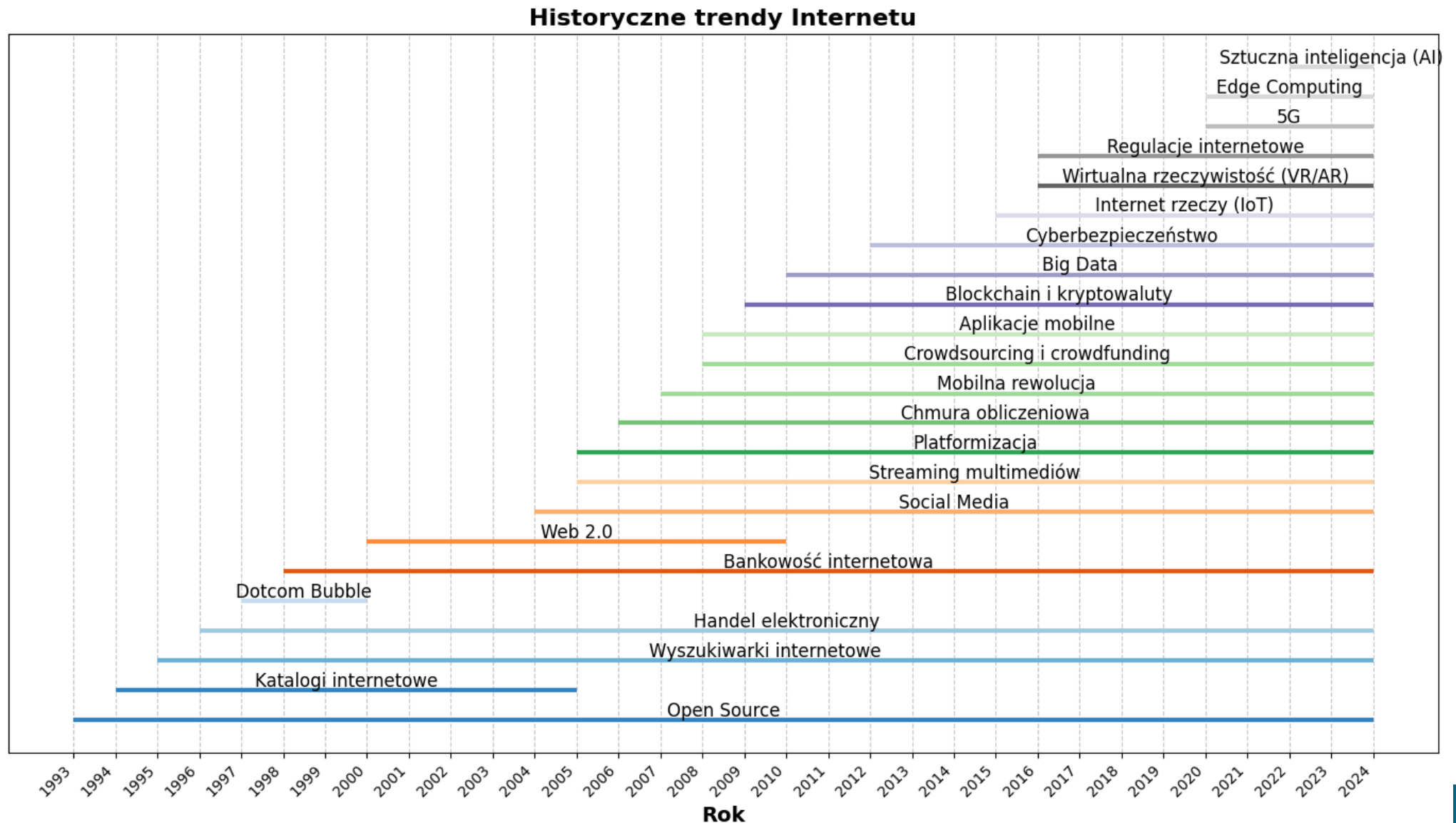


Sędzia

Full-Stack



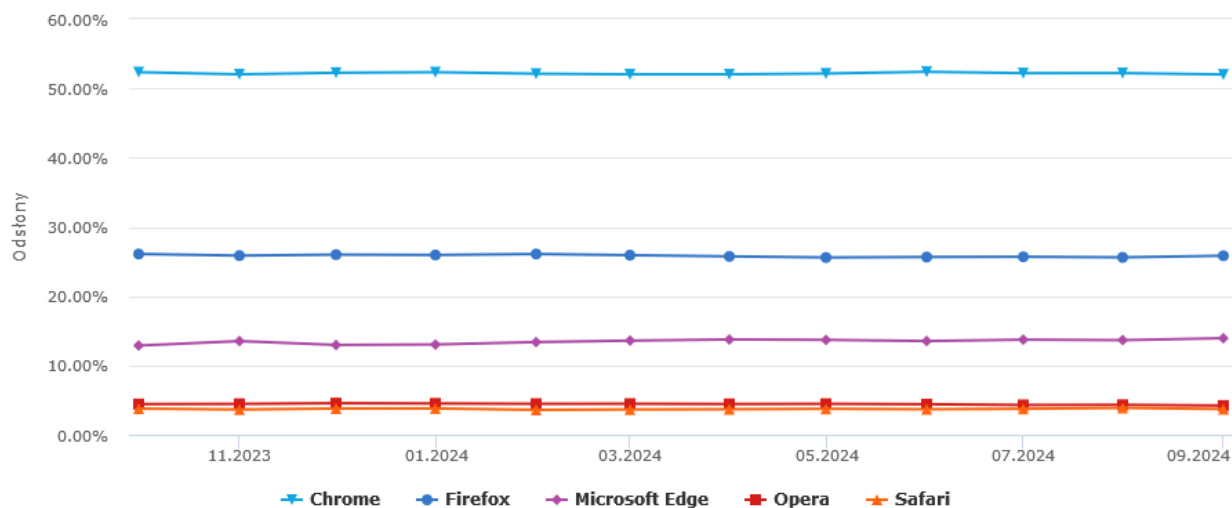
Trendy Internetu - ostatnie 30 lat



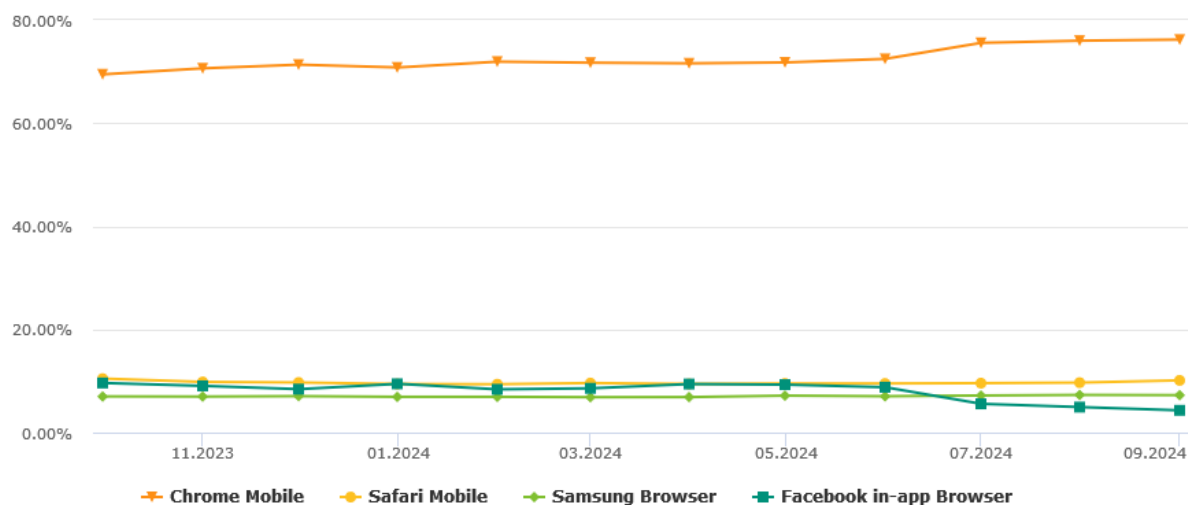
Popularność przeglądarek



PRZEGLĄDARKI - GRUPY
(POLSKA, PC)



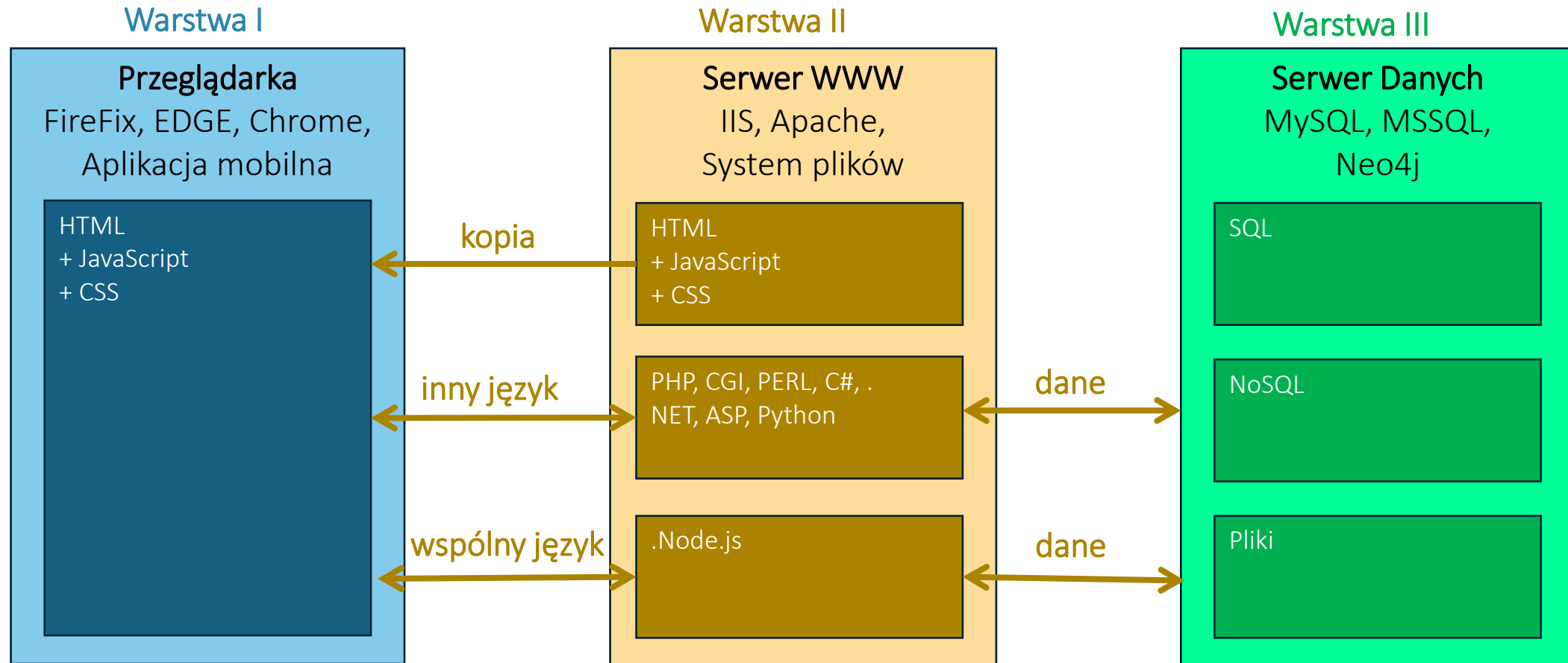
PRZEGLĄDARKI - GRUPY
(POLSKA, TELEFON)



Gemius Ranking

<https://ranking.gemius.com/pl/>

Architektura aplikacji WWW



GET, POST

GET

służy do pobierania danych z serwera.

Dane te są zazwyczaj zwracane w formacie HTML, JSON lub XML i mogą być wyświetlane użytkownikowi w przeglądarce internetowej.



POST

służy do wysyłania danych do serwera w celu wykonania określonej akcji.

Dane te mogą być formularzami, plikami lub dowolnymi innymi danymi, które mają zostać przetworzone przez serwer.



GET, POST

GET vs POST

Aspekt	GET	POST
Wizualizacja danych	Parametry są dołączane do URL, co sprawia, że są widoczne dla użytkownika.	Dane są wysyłane w treści żądania, zwykle nie są widoczne dla użytkownika.
Ograniczenia długości	Ograniczona długość URL przez przeglądarkę, co wpływa na ilość danych.	Teoretycznie brak ograniczeń długości, może przysyłać dużo większe ilości danych.
Zapisywanie w cache	Możliwość zapisywania w pamięci cache.	Zwykle odpowiedzi nie są cachowane.
Bezpieczeństwo	Mniej bezpieczny, ponieważ dane są wyświetlane w URL.	Bezpieczniejszy dla poufnych danych, ponieważ nie są wyświetlane w URL.
Historia przeglądarki	Parametry pozostają w historii przeglądarki.	Dane nie pozostają w historii przeglądarki.
Typy danych	Ograniczone głównie do tekstowych danych zapytania.	Może przysyłać różnorodne typy danych, w tym binarne.
Zastosowanie	Pobieranie danych, np. wyszukiwanie, odświeżanie treści. Operacje, które mogą być powtarzane bez skutków ubocznych. Żądania, które mogą być zakodowane w linku (np. zakładki).	Wysyłanie danych do serwera, np. przysyłanie formularzy. Operacje, które powinny być wykonywane tylko raz, np. złożenie zamówienia. <u>Przesyłanie dużych lub poufnych danych.</u>

jQuery

jQuery (JavaScript Queries) <https://jquery.com/>

to biblioteka JavaScript, która upraszcza manipulację DOM, obsługę zdarzeń, animacje oraz komunikację Ajax, umożliwiając szybsze i łatwiejsze tworzenie interaktywnych stron internetowych.

Pozwala na dynamiczne zarządzanie kontentem i zachowaniami na stronie.



Manipulacja
DOM

Obsługa
zdarzeń

Efekty i
animacje

Ajax

jQuery

Dołączenie biblioteki jQuery
załadowanie biblioteki jQuery
z CDN (Content Delivery
Network) Google.

Metoda `$(document).ready()`
używana do wykonania funkcji
JavaScript dopiero po
załadowaniu całego drzewa
DOM strony

Selektor `$("p")` i metoda `.slideDown()`
wybiera element `<p>` i automatycznie
przy ładowaniu strony aplikuje efekt
`slideDown()`, który rozwija element
pionowo, tworząc animację
rozwijania.
Opcja `'slow'` określa, że animacja
powinna być wykonana wolno

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Przykład jQuery</title>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
<body>
  <h1>Witaj w świecie jQuery!</h1>
  <p style="display: none;">Jestem paragrafem ukrytym przed załadowaniem.</p>

  <script>
    $(document).ready(function() {
      // Efekt fadeIn dla elementu h1 po kliknięciu
      $("h1").click(function() {
        $(this).fadeIn('slow');
      });

      // Efekt slideDown dla elementu p przy ładowaniu strony
      $("p").slideDown('slow');
    });
  </script>
</body>
</html>
```

Stylowanie CSS w elemencie `<p>`
element `<p>` ma zastosowany styl
CSS `display: none;`, co oznacza, że
jest on ukryty na stronie do czasu
wywołania przez jQuery efektu
`slideDown()`

Selektor `$("h1")` i metoda `.click()`
wybiera element HTML `<h1>` i przypisuje
mu funkcję, która będzie wywoływana
przy każdym kliknięciu na ten element

Metody `.fadeIn()`
do płynnego pokazania elementu z efektem
przeźroczystości.

jQuery

Metody dodania

1. Dołączanie jQuery z CDN (Content Delivery Network):
 - Google CDN
 - Microsoft CDN
 - CDNJS Cloudflare
2. Lokalne hostowanie jQuery:
 - Pobierz minifikowaną wersję jQuery z oficjalnej strony jQuery.
 - Umieść plik w katalogu swojego projektu.
 - Dołącz plik w swoim pliku HTML
- Dołączanie jQuery w środowisku Node.js:
 - Instalacja jQuery za pomocą npm (npm install jquery)
 - Instalacja jQuery za pomocą yarn (yarn add jquery)
 - Importowanie jQuery w skryptach JavaScript używając require lub import

```
<!-- Dołączanie najnowszej wersji jQuery z Google CDN -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<!-- Dołączanie jQuery z Microsoft CDN -->
<script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.5.1.min.js"></script>

<!-- Dołączanie jQuery z CDNJS Cloudflare -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

```
<script src="ścieżka/do/pliku/jquery-3.5.1.min.js"></script>
```

jQuery

Selektory

pozwalają na szybkie i efektywne przeszukiwanie drzewa DOM w celu znalezienia konkretnych elementów, z którymi chcemy pracować.

Można je wykorzystać do zmiany stylów, zawartości, obsługi zdarzeń i wiele innych.

Przykłady Selektorów

`$('div')` - wybiera wszystkie elementy po tagu.

`$('.klasa')` - wybiera wszystkie elementy o danej klasie.

`$('#id')` - wybiera element o określonym ID.

`$('ul li:first-child')` - wybiera pierwsze dzieci elementów `<li>` w każdym `<ul>`.

```
<div>Div 1</div>
<div>Div 2</div>
<div class="klasa">Element z klasą 'klasa'</div>
<div id="unikalneID">Element z unikalnym ID</div>
<ul><li>Pierwszy element listy</li><li>Drugi element listy</li></ul>
<ul><li>Kolejny pierwszy element</li><li>Kolejny drugi element</li></ul>

<script>
  $(document).ready(function() {
    $('div').css('border', '1px solid blue');
    $('.klasa').css('font-weight', 'bold');
    $('#unikalneID').css('text-decoration', 'underline');
    $('ul li:first-child').css('background-color', 'yellow');
  });
</script>
```

jQuery

Modyfikacja zawartości

.html(): Zmienia zawartość HTML wewnątrz wybranych elementów.

.text(): Zmienia tekstową zawartość wybranych elementów, usuwając wszelki HTML.

Modyfikacja struktury dokumentu

.remove(): Usuwa wybrane elementy (wraz z ich potomkami) z dokumentu.

.empty(): Usuwa wszystkie potomne elementy wewnątrz wybranych elementów, pozostawiając same puste kontenery

```
<div id="content">Oryginalna zawartość</div>
<button id="changeContent">Zmień zawartość</button>
<button id="changeText">Zmień tekst</button>
<button id="removeElement">Usuń element</button>
<button id="emptyElement">Opróżnij element</button>

<script>
$(document).ready(function() {
    $('#changeContent').click(function() {
        $('#content').html('<p>Zmieniona zawartość HTML</p>');
    });
    $('#changeText').click(function() {
        $('#content').text('Zmieniony tekst, bez HTML');
    });
    $('#removeElement').click(function() {
        $('#content').remove();
    });
    $('#emptyElement').click(function() {
        $('#content').empty();
    });
});
</script>
```

jQuery

Reakcja na interakcje użytkownika

.show(): Natychmiastowo pokazuje ukryty element, sprawiając, że staje się widoczny.

.hide(): Natychmiastowo ukrywa element, czyniąc go niewidocznym dla użytkownika.

.toggle(): Przełącza stan widoczności elementu — jeśli jest ukryty, pokazuje go; jeśli widoczny, ukrywa.

.fadeIn(): Stopniowo zwiększa przezroczystość ukrytego elementu, aż stanie się on w pełni widoczny, tworząc efekt pojawiania się.

.fadeOut(): Stopniowo zmniejsza przezroczystość elementu do całkowitego zniknięcia, tworząc efekt znikania.

.slideDown(): Stopniowo "rozsuka" element, odsłaniając go i tworząc efekt rozwijania się z góry do dołu.

.slideUp(): Stopniowo "zsuwa" element, ukrywając go i tworząc efekt zwijania się z dołu do góry

```
<html lang="pl">
<head>
<meta charset="UTF-8">
<title>Przykłady jQuery</title>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<style>
.box { width: 100px; height: 100px; background-color: #007bff; margin: 10px 0; display: none; }
</style>
</head>
<body>

<div id="box" class="box"></div>

<button onclick="$('#box').show()">Show</button>
<button onclick="$('#box').hide()">Hide</button>
<button onclick="$('#box').toggle()">Toggle</button>
<button onclick="$('#box').fadeIn()">Fade In</button>
<button onclick="$('#box').fadeOut()">Fade Out</button>
<button onclick="$('#box').slideDown()">Slide Down</button>
<button onclick="$('#box').slideUp()">Slide Up</button>

</body>
</html>
```

Ajax

Ajax (Asynchronous JavaScript and XML)

to technika umożliwiająca asynchroniczne przesyłanie danych między klientem a serwerem, co pozwala na aktualizację części strony internetowej bez konieczności przeładowywania całej strony.



Metoda	Typ Żądania	Prostota	Kontrola	Zastosowanie typowe
.ajax()	Dowolny	Niska	Pełna	Skomplikowane zapytania, konfiguracja nagłówków, obsługa różnych formatów danych, aplikacje SPA, zaawansowane interakcje z REST API.
.get()	GET	Wysoka	Ograniczona	Proste pobieranie danych (tekst, JSON), szybkie aktualizacje bez konieczności pełnej konfiguracji.
.post()	POST	Wysoka	Ograniczona	Wysyłanie danych, takich jak formularze i informacje użytkownika, zwykle przy przesyłaniu danych do serwera.
.load()	GET	Średnia	Ograniczona	Dynamiczne ładowanie treści HTML do określonej części strony, aktualizacje sekcji bez przeładowywania całej strony.

Ajax

Metoda \$.ajax()

Uniwersalna metoda do tworzenia żądań Ajax, oferująca pełną kontrolę nad konfiguracją żądania.

```
$.ajax({  
  url: 'ścieżka/do/zasobu',    // URL żądania  
  type: 'GET',                // Metoda HTTP, 'GET' lub 'POST'  
  dataType: 'json',           // Oczekiwany typ danych odpowiedzi, np. 'json', 'xml', 'html', itp.  
  data: {                     // Dane wysyłane z żądaniem  
    klucz1: 'wartość1',  
    klucz2: 'wartość2'  
  },  
  success: function(data) {    // Funkcja wywoływana w przypadku sukcesu  
    // Obsługa danych 'data' zwróconych z serwera  
  },  
  error: function(jqXHR, textStatus, errorThrown) { // Funkcja wywoływana w przypadku błędu  
    // Obsługa błędu  
  }  
});
```

```
<html lang="pl">  
<head>  
  <meta charset="UTF-8">  
  <title>AJAX w jQuery</title>  
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
  <script>  
    $(function() {  
      $('#ajaxRequestButton').on('click', function() {  
        $.ajax({  
          url: 'ścieżka/do/zasobu',  
          type: 'GET',  
          dataType: 'json',  
          success: function(data) {  
            $('#responseContainer').text(JSON.stringify(data));  
          },  
          error: function() {  
            $('#responseContainer').text('Błąd żądania');  
          }  
        });  
      });  
    });  
  </script>  
</head>  
<body>  
  
  <button id="ajaxRequestButton">Wyślij żądanie AJAX</button>  
  <div id="responseContainer"></div>  
  
</body>  
</html>
```

Ajax

Metoda \$.get()

Metoda do wysyłania żądań GET.

```
$.get('ścieżka/do/zasobu',    // URL żądania
    {                        // Dane wysyłane z żądaniem
      klucz1: 'wartość1',
      klucz2: 'wartość2'
    },
    function(data) {          // Funkcja wywoływana w przypadku sukcesu
      // Obsługa danych 'data' zwróconych z serwera
    },
    'json'                    // Oczekiwany typ danych odpowiedzi, np. 'json'
).fail(function(jqXHR, textStatus, errorThrown) { // Funkcja wywoływana w przypadku błędu
  // Obsługa błędu
});
```

```
<html lang="pl">
<head>
<meta charset="UTF-8">
<title>Przykład użycia jQuery .get()</title>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
$(document).ready(function(){
  $('#ajaxRequestButton').click(function(){
    $.get('ścieżka/do/zasobu', { klucz1: 'wartość1', klucz2: 'wartość2' }, function(data) {
      $('#responseContainer').text(JSON.stringify(data));
    }, 'json')
    .fail(function(jqXHR, textStatus, errorThrown) {
      $('#responseContainer').text('Wystąpił błąd: ' + textStatus);
    });
  });
});
</script>
</head>
<body>

<button id="ajaxRequestButton">Wyślij żądanie GET</button>
<div id="responseContainer">Odpowiedź serwera zostanie wyświetlona tutaj.</div>

</body>
</html>
```

Ajax

Metoda \$.load()

Służy do ładowania danych z określonego URL-a i wstawiania ich bezpośrednio do wybranego elementu DOM.

```
$('#elementId').load('ścieżka/do/zasobu', // URL żądania
{ // Dane wysyłane z żądaniem
  klucz1: 'wartość1',
  klucz2: 'wartość2'
},
function(responseText, textStatus, xhr) { // Funkcja wywoływana po zakończeniu żądania
  if (textStatus == "error") {
    // Obsługa błędu, np. wyświetlenie komunikatu
  } else {
    // Obsługa danych 'responseText' zwróconych z serwera
  }
});
```

```
<html lang="pl">
<head>
<meta charset="UTF-8">
<title>Przykład jQuery .load()</title>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
$(document).ready(function(){
  // Przykładowe wywołanie może być zainicjowane przez zdarzenie, np. kliknięcie przycisku.
  $('#loadDataButton').click(function() {
    $('#elementId').load('ścieżka/do/zasobu', {
      klucz1: 'wartość1',
      klucz2: 'wartość2'
    }, function(responseText, textStatus, xhr) {
      if (textStatus == "error") {
        $('#elementId').text('Wystąpił błąd: ' + xhr.status + " " + xhr.statusText);
      } else {
        // Tutaj 'responseText' zawiera treść załadowaną z 'ścieżka/do/zasobu'
        // Jeśli jest potrzebna dodatkowa logika, można ją dodać tutaj.
      }
    });
  });
});
</script>
</head>
<body>

<button id="loadDataButton">Załaduj dane</button>
<div id="elementId">Tu pojawią się załadowane dane.</div>

</body>
</html>
```

Ajax

Metoda \$.post()

Metoda do wysyłania żądań POST.

```
$.post('ścieżka/do/zasobu', // URL żądania
{ // Dane wysyłane z żądaniem
  klucz1: 'wartość1',
  klucz2: 'wartość2'
},
function(data) { // Funkcja wywoływana w przypadku sukcesu
  // Obsługa danych 'data' zwróconych z serwera
},
'json' // Oczekiwany typ danych odpowiedzi, np. 'json'
).fail(function(jqXHR, textStatus, errorThrown) { // Funkcja wywoływana w przypadku błędu
  // Obsługa błędu
});
```

```
<html lang="pl">
<head>
<meta charset="UTF-8">
<title>Przykład z jQuery .post()</title>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
$(document).ready(function(){
  $('#ajaxRequestButton').click(function(){
    $.post('ścieżka/do/zasobu', { klucz1: 'wartość1', klucz2: 'wartość2' }, function(data) {
      $('#responseContainer').text(JSON.stringify(data));
    }, 'json')
    .fail(function(jqXHR, textStatus, errorThrown) {
      $('#responseContainer').text('Wystąpił błąd: ' + textStatus);
    });
  });
});
</script>
</head>
<body>

<button id="ajaxRequestButton">Wyślij żądanie POST</button>
<div id="responseContainer">Odpowiedź serwera zostanie wyświetlona tutaj.</div>

</body>
</html>
```

Synchroniczność, Asynchroniczność

Synchroniczność

oznacza wykonywanie operacji sekwencyjnie, jedna po drugiej.

Oznacza to, że program musi zakończyć jedną operację, zanim rozpocznie inną.



Asynchroniczność

oznacza wykonywanie operacji niezależnie od siebie.

Oznacza to, że program nie musi czekać na zakończenie jednej operacji, aby rozpocząć inną.



Synchroniczność, Asynchroniczność

Synchroniczność vs Asynchroniczność

Aspekt	Synchroniczność	Asynchroniczność
Zastosowanie	Sekwencyjne operacje; prostsze zadania i skrypty.	Operacje I/O; interaktywne aplikacje; zadania w tle.
Zalety	Prosta logika przepływu; łatwa obsługa błędów.	Nie blokuje interfejsu użytkownika; wydajność; skalowalność.
Wady	Może blokować UI; nieefektywne dla operacji I/O.	Trudniejsza obsługa błędów; złożoność w zarządzaniu stanem.
Responsywność UI	Niska podczas długotrwałych operacji.	Wysoka, ponieważ UI jest dostępny podczas przetwarzania.
Wydajność	Może być niska dla wielu jednoczesnych operacji I/O.	Wysoka, pozwala na jednoczesne przetwarzanie wielu zadań.
Skalowalność	Ograniczona w zależności od czasu trwania operacji.	Wysoka, umożliwia obsługę wielu żądań jednocześnie.
Obsługa błędów	Bezpośrednia; sekwencyjna w miejscu wystąpienia.	Może wymagać złożonych mechanizmów.
Przepływ kontroli	Linearny i przewidywalny.	Może być nieliniarny i wymagać szczególnego zarządzania.

UI – interfejs użytkownika

operacji I/O (InPut/OutPut) – operacje wejścia/wyjścia

Synchroniczność, Asynchroniczność

Synchroniczność w jQuery

jest obecnie rzadko używane i generalnie uznawane za złą praktykę ze względu na negatywny wpływ na UI.

Ustawienie async: false prowadzi do synchronicznego żądania

```
$.ajax({  
  url: 'ścieżka/do/zasobu',  
  async: false, // Niezalecane, używaj z ostrożnością!  
  success: function(data) {  
    // obsługa danych  
  }  
});
```

Asynchroniczność w jQuery

jest standardem w jQuery i w programowaniu webowym.

Ustawienie async: true prowadzi do asynchronicznego żądania i jest to wartość domyślna

```
$.ajax({  
  url: 'ścieżka/do/zasobu',  
  success: function(data) {  
    // obsługa danych  
  }  
});
```

Dotyczy to metod \$.ajax, \$.get, \$.post, \$.load

Cookie

Cookie

zwane również ciasteczkami, to małe pliki tekstowe przechowywane na komputerze lub urządzeniu użytkownika przez przeglądarkę internetową podczas odwiedzania stron internetowych. Pliki te zawierają dane dotyczące użytkownika i jego preferencji, co umożliwia stronom internetowym pamiętanie informacji o użytkowniku podczas przyszłych wizyt.

Standardowa specyfikacja HTTP RFC 6265 określa, że rozmiar pojedynczego cookie nie może przekraczać 4 KB (4096 bajtów).

Przeglądarki zwykle pozwalają na przechowywanie około 50 cookies na jedną domenę.

Całkowity limit cookies dla jednej przeglądarki wynosi około 3000 cookies.

```
name=value; expires=Wed, 21 Oct 2023 07:28:00 GMT; path=/; domain=example.com; secure; HttpOnly
```

Cookie

Parametry cookie

Atrybut	Opis
value	Wartość ciasteczka w postaci klucz=wartość. To informacje, które są faktycznie przechowywane i dostępne podczas sesji.
expires	Data wygaśnięcia ciasteczka. Po tej dacie ciasteczko przestaje być dostępne.
max-age	Maksymalny czas życia ciasteczka w sekundach od momentu utworzenia.
domain	Domena, dla której ciasteczko jest dostępne. Ciasteczko będzie dostępne dla subdomen, jeśli domena jest ustawiona na domenę wyższego poziomu.
path	Ścieżka na serwerze, dla której ciasteczko będzie dostępne. Węższa ścieżka ogranicza dostępność ciasteczka tylko do tej ścieżki.
secure	Gdy atrybut jest ustawiony, ciasteczko jest wysyłane tylko z zapytaniem przesyłanym przez HTTPS.
HttpOnly	Kiedy ten atrybut jest ustawiony, ciasteczko nie jest dostępne przez skrypty klienta (np. JavaScript), co pomaga zapobiegać atakom typu XSS.

Cookie

Zastosowania Cookies

Autoryzacja:

- Umożliwiają utrzymanie sesji użytkownika po zalogowaniu.

Analiza:

- Pomagają właścicielom stron internetowych analizować ruch na stronie i zachowania użytkowników.

Reklama:

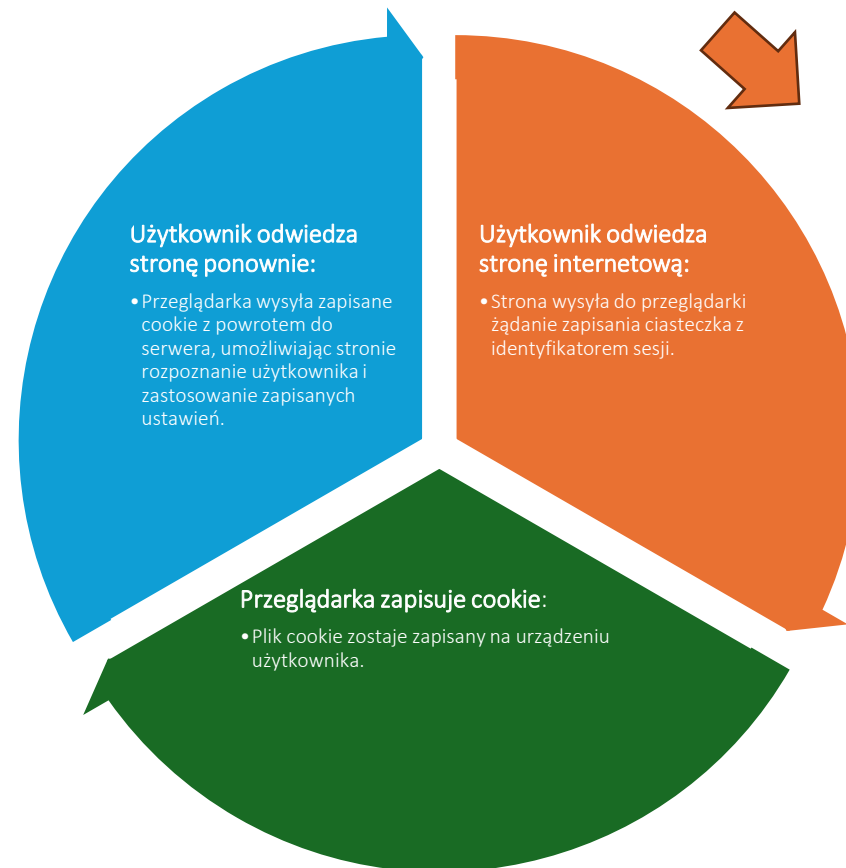
- Umożliwiają reklamodawcom dostosowanie reklam do zainteresowań użytkowników.

Personalizacja:

- Przechowują preferencje użytkownika, takie jak ustawienia języka czy układ strony.

Cookie

Przykład Działania Cookie



MIME

MIME (Multipurpose Internet Mail Extensions)

to standard, który rozszerza formaty poczty elektronicznej do obsługi tekstu w zestawach znaków innych niż ASCII, a także załączników multimedialnych (grafika, audio, wideo).

MIME jest powszechnie używany w protokole HTTP, który jest podstawą komunikacji w Internecie, aby określić typy danych przesyłanych między klientem a serwerem.

Struktura typu MIME

- **Typ główny (primary type):** określa ogólną kategorię danych (np. text, image, audio, video, application).
- **Podtyp (subtype):** określa bardziej szczegółowo typ danych (np. text/html, image/png, application/json).

```
header("Content-Type: application/json");  
echo json_encode($data);
```

```
const express = require('express');  
const app = express();  
  
app.get('/json', (req, res) => {  
  res.type('application/json');  
  res.send({ name: "John Doe", email: "john.doe@example.com" });  
});  
  
app.listen(3000, () => {  
  console.log('Server is running on port 3000');  
});
```

MIME

Typy MIME

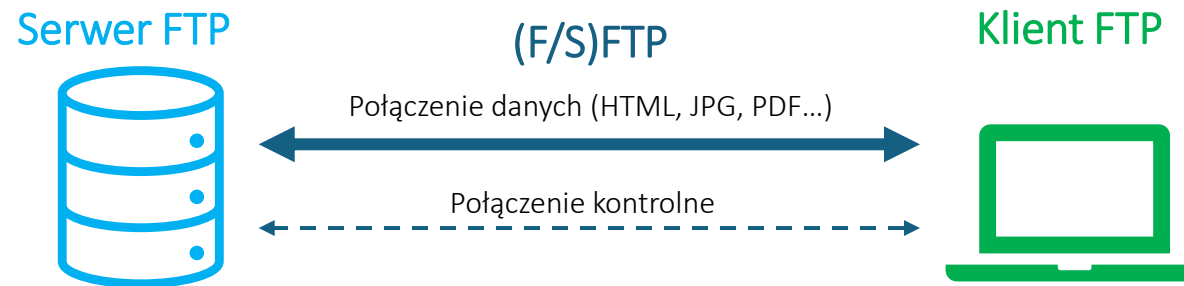
Typ główny	Opis	Podtypy
text	Przeznaczony do przesyłania treści tekstowych.	- text/plain: zwykły tekst - text/html: dokumenty html - text/css: arkusze stylów css - text/javascript: skrypty javascript - application/javascript: skrypty javascript
image	Przeznaczony do przesyłania obrazów.	- image/jpeg: obrazy jpeg - image/png: obrazy png - image/gif: obrazy gif - image/svg+xml: obrazy svg
audio	Przeznaczony do przesyłania dźwięku.	- audio/mpeg: pliki audio MP3 - audio/ogg: pliki audio ogg
video	Przeznaczony do przesyłania wideo.	- video/mp4: pliki wideo MP4 - video/ogg: pliki wideo OGG
application	Przeznaczony do przesyłania danych specyficznych dla aplikacji.	- application/json: dane w formacie JSON - application/xml: dane w formacie XML - application/pdf: dokumenty PDF - application/zip: skompresowane pliki ZIP

FTP

Protokół FTP (File Transfer Protocol)

to protokół sieciowy stosowany do przesyłania plików między klientem a serwerem w sieci komputerowej.

FTP jest jednym z najstarszych protokołów internetowych i jest szeroko używany do przesyłania plików na serwery internetowe, zarządzania stronami internetowymi, oraz do pobierania i udostępniania plików.



Standardowy protokół FTP nie szyfruje danych, co oznacza, że hasła i pliki przesyłane przez sieć są podatne na przechwycenie przez osoby trzecie.

Wersje szyfrowane:

- **FTPS (FTP Secure):** Rozszerzenie FTP, które obsługuje SSL/TLS do szyfrowania połączeń.
- **SFTP (SSH File Transfer Protocol):** Bezpieczny protokół transferu plików.

Uwierzytelnienie

Uwierzytelnienie

to proces weryfikacji tożsamości użytkownika lub systemu, który próbuje uzyskać dostęp do zasobów.

W kontekście technologii internetowych istnieje kilka mechanizmów uwierzytelniania, które zapewniają różne poziomy bezpieczeństwa.

Cechy

Bezpieczeństwo

- Ochrona danych uwierzytelniających przed nieuprawnionym dostępem i przechwyceniem, stosowanie silnych metod szyfrowania i technik zabezpieczających.

Użyteczność

- Łatwość korzystania z systemu uwierzytelniania; proces powinien być intuicyjny i wygodny, aby użytkownicy nie szukali mniej bezpiecznych metod dostępu.

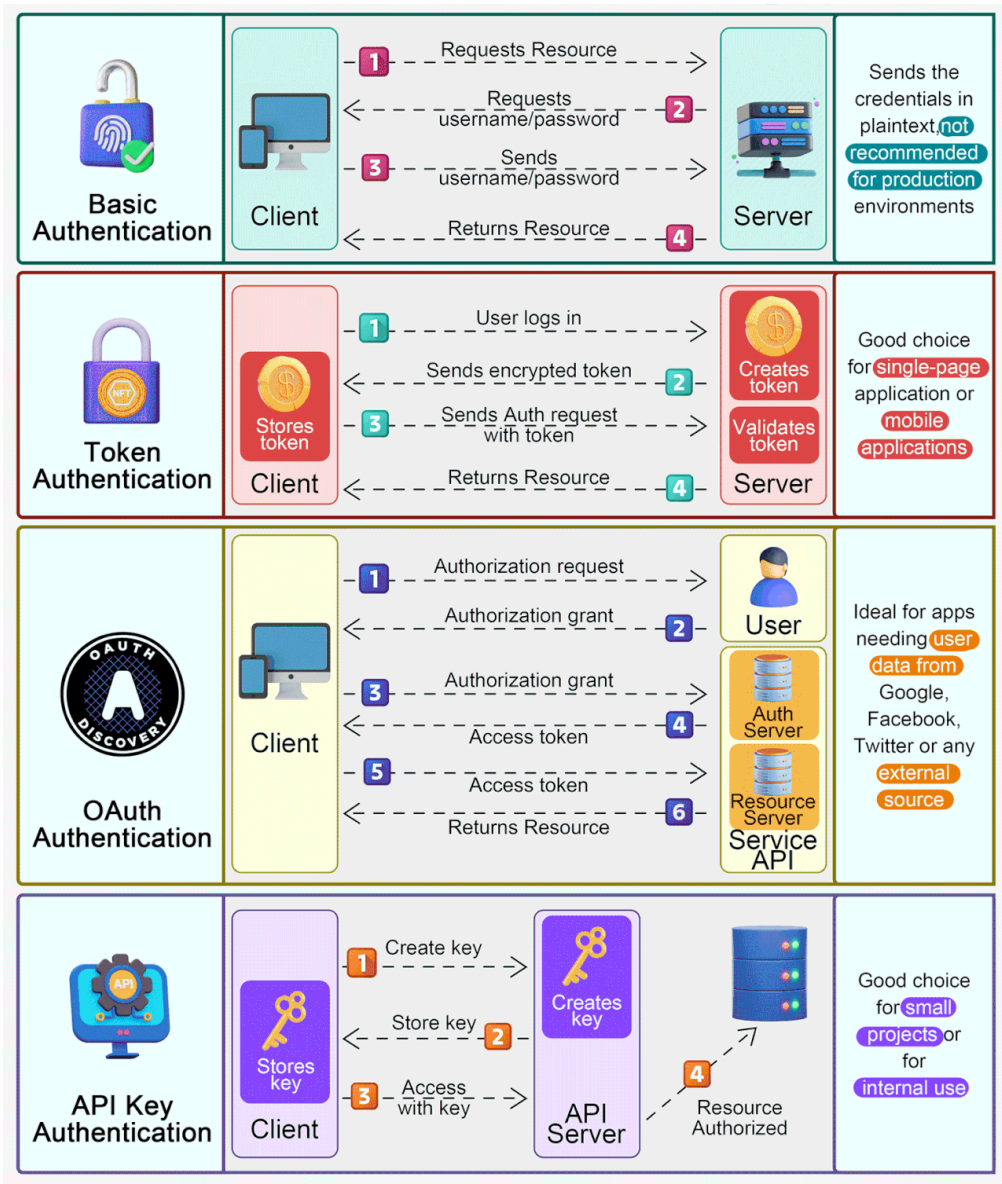
Niezawodność

- Działanie systemu uwierzytelniania zgodnie z przeznaczeniem, bez awarii i błędów, zapewniając ciągły dostęp użytkownikom.

Skalowalność

- Zdolność systemu do obsługi rosnącej liczby użytkowników bez pogorszenia wydajności, możliwość rozszerzania możliwości w miarę wzrostu liczby użytkowników i złożoności operacji.

Uwierzytelnienie



Metoda	Zalety	Wady	Zastosowanie
Basic Authentication	- Prosta implementacja. - Powszechnie obsługiwana.	- Przesyłanie danych w formie zwykłego tekstu (plaintext). - Niezalecane w środowiskach produkcyjnych.	Dobre rozwiązanie dla małych projektów lub środowisk wewnętrznych.
Token Authentication	- Bezpieczniejsze niż podstawowe uwierzytelnianie. - Token może być używany wielokrotnie do czasu wygaśnięcia.	- Konieczność zarządzania i przechowywania tokenów.	Idealne dla aplikacji jednowarstwowych (SPA) i aplikacji mobilnych.
OAuth Authentication	- Bezpieczne zarządzanie dostępem do zasobów. - Umożliwia dostęp do zasobów z różnych usług zewnętrznych.	- Może być skomplikowane do implementacji. - Wymaga zaufanych serwerów autoryzacyjnych.	Idealne dla aplikacji korzystających z danych użytkowników z serwisów zewnętrznych.
API Key Authentication	- Prosta implementacja. - Klucze API mogą być łatwo zarządzane i rotowane.	- Klucze API mogą być narażone na kradzież. - Nie zapewnia zaawansowanego uwierzytelniania użytkowników.	Dobre rozwiązanie dla małych projektów lub wewnętrznych systemów.

**THE MORE YOU LEARN,
THE MORE YOU EARN.**

@successpictures

