



dr inż. Michał Malinowski
michal.malinowski@warszawa.merito.pl



[https://www.linkedin.com/in/
michał-malinowski-malkon/](https://www.linkedin.com/in/michał-malinowski-malkon/)

Manipulacja DOM

Wykład 03

JĘZYKI PROGRAMOWANIA
STUDIUM PRZYPADKU



Zagadnienia (2h)

- Full-Stack
- Architektura aplikacji WWW
- Hierarchia obiektów JavaScript
- DOM (Document Object Model)
 - navigator
 - Array
 - Date
 - math
 - regExp
 - window
 - document



***Musimy myśleć o
programowaniu jak
o fundamentalnej
umiejętności w XXI w.
Porównałbym to do
czytania, pisania, liczenia.***

Września 2023

Miron Mironiuk, założyciel Cosmose.AI,
pomysłodawca programu "Programowanie = nasz drugi język"

Full-Stack

Full-Stack (Deweloper)

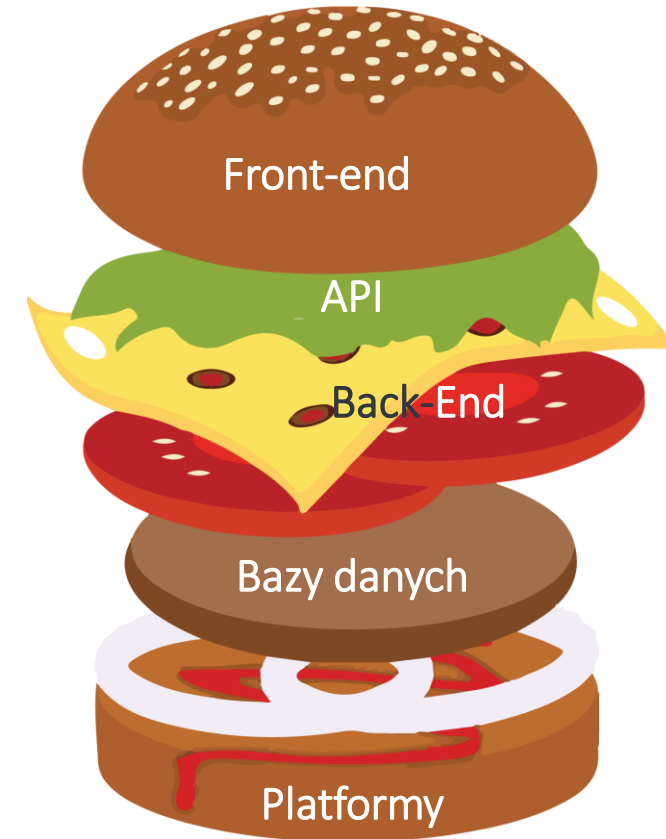
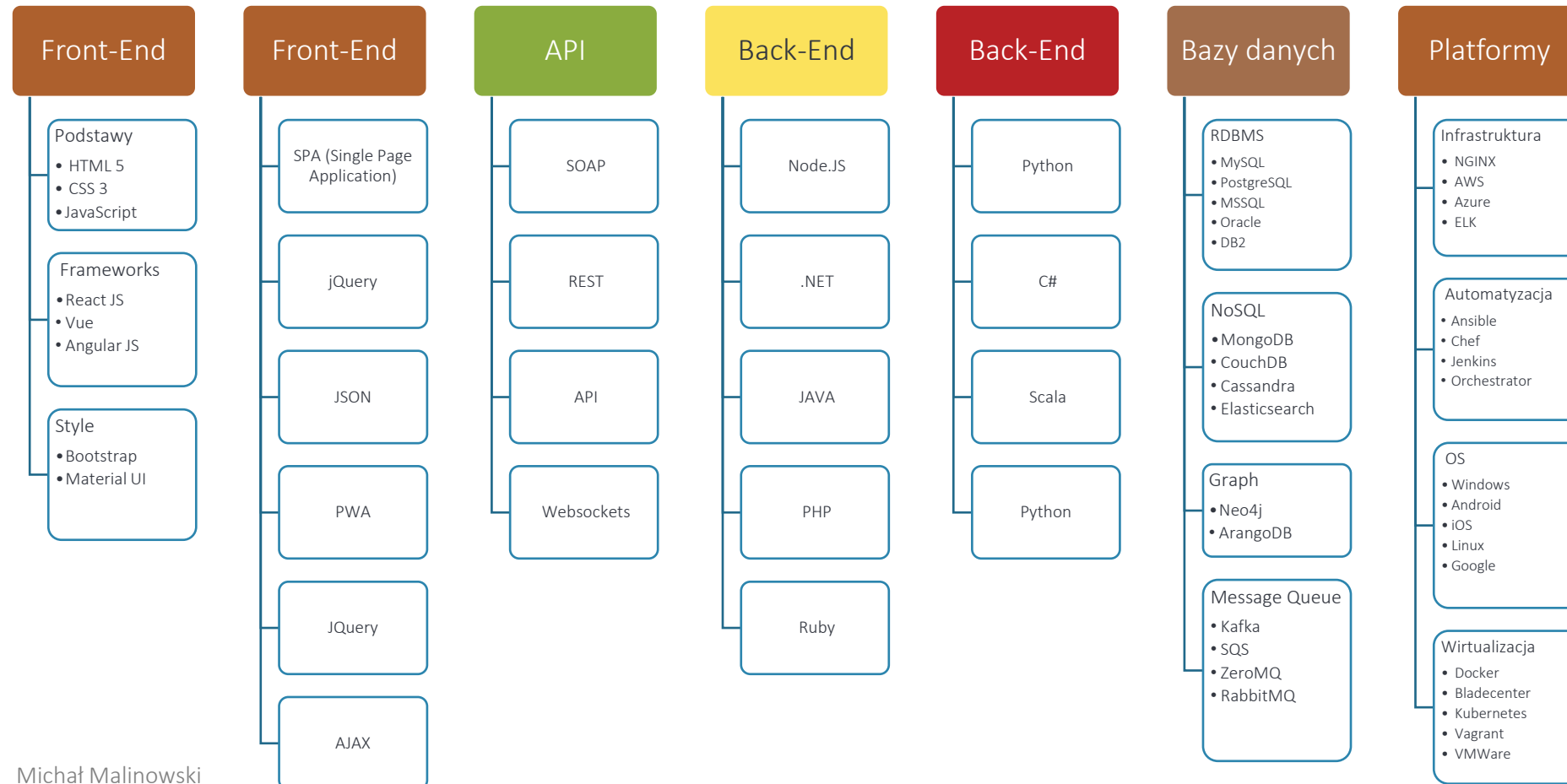
odnosi się do kompletu technologii i umiejętności niezbędnych do projektowania, implementacji oraz zarządzania całą strukturą aplikacji internetowej lub mobilnej.

Obejmuje to zarówno

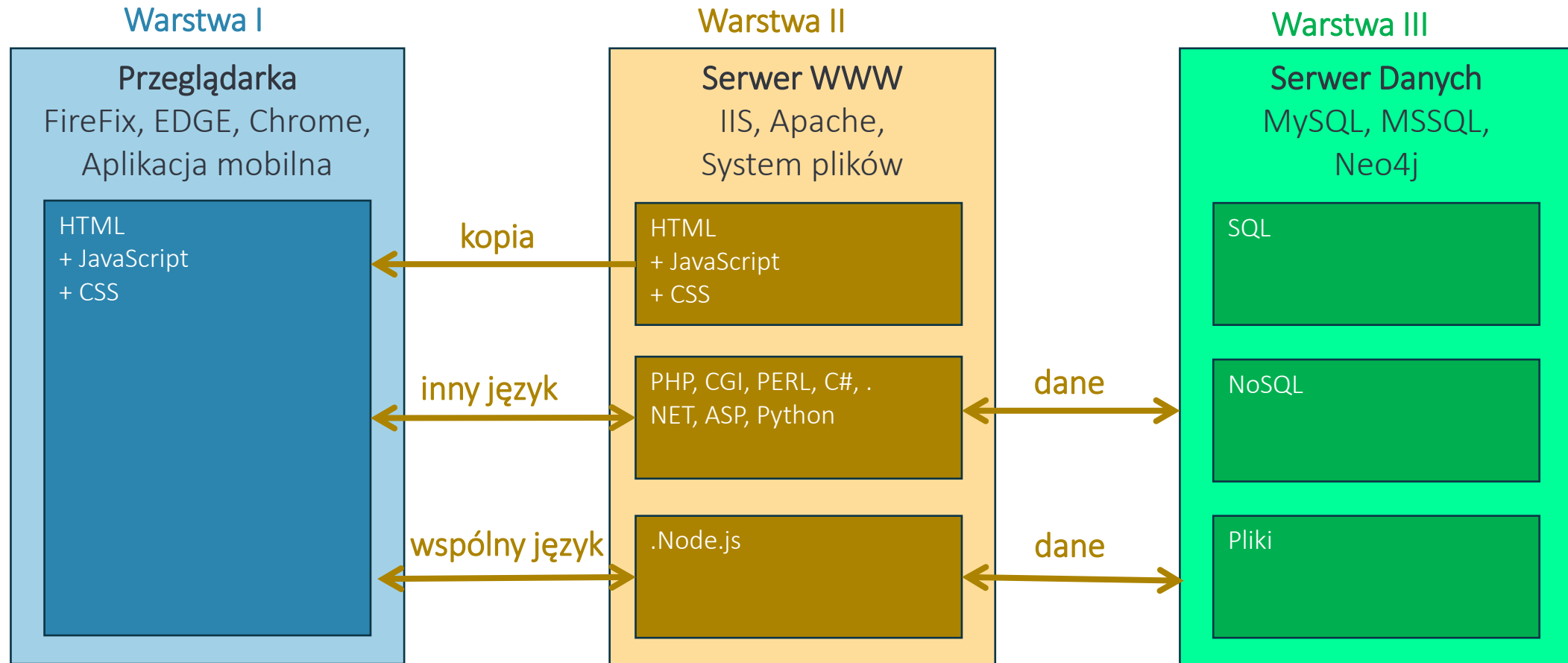
- Front-End, czyli część aplikacji widoczną i dostępną dla użytkownika (interfejs użytkownika),
- Back-End, czyli część serwerową odpowiedzialną za logikę działania aplikacji, przetwarzanie danych i komunikację z bazą danych.

Programista Full-Stack jest więc w stanie samodzielnie stworzyć pełną aplikację webową lub mobilną, od zera do wersji produkcyjnej.

Full-Stack

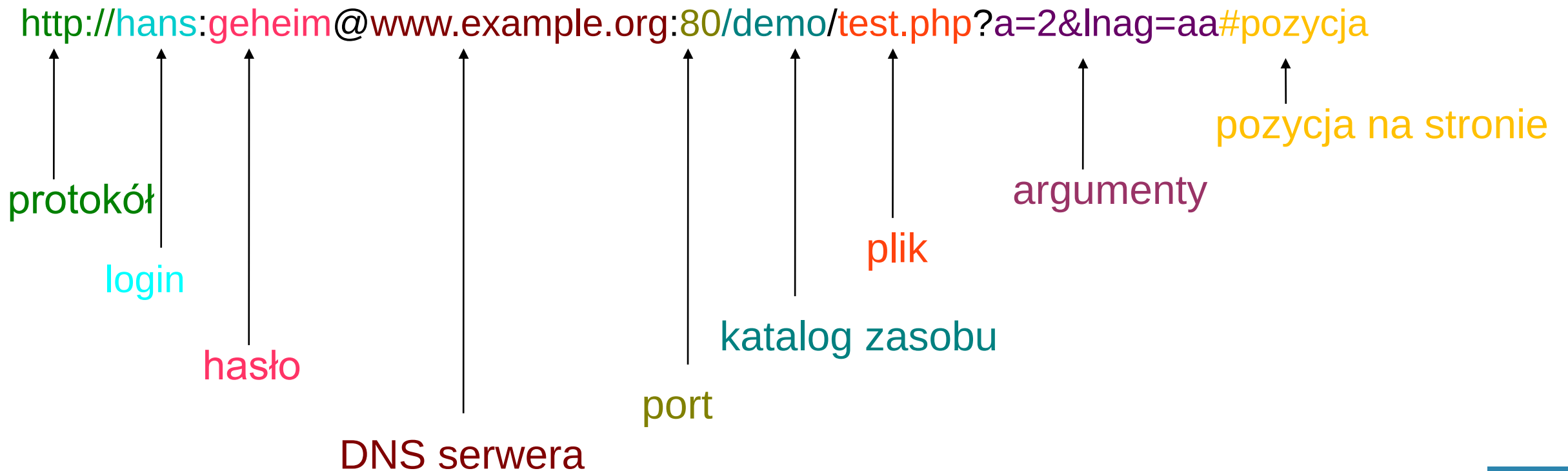


Architektura aplikacji WWW



Architektura aplikacji WWW

Schemat struktury adresu URL (Uniform Resource Locator)



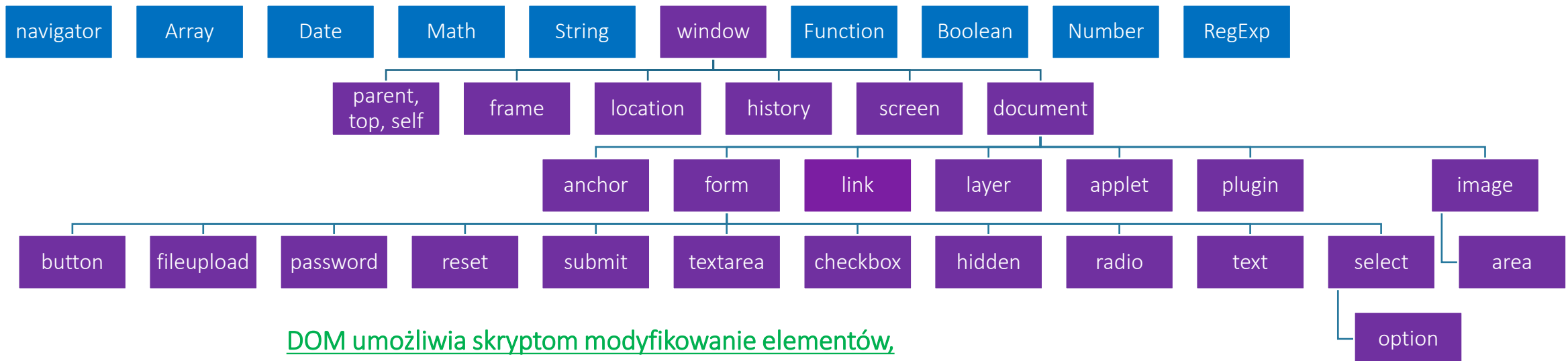
Hierarchia obiektów JavaScript

Hierarchia obiektów JavaScript (JavaScript Object Hierarchy)

jest to struktura danych, w której obiekty są połączone ze sobą relacjami rodzic-dziecko. Pozwala to na organizowanie i dostęp do danych w uporządkowany sposób.

DOM (Document Object Model)

jest to interfejs programowania aplikacji (API) definiujący sposób, w jaki JavaScript może manipulować strukturą i zawartością dokumentu HTML.



DOM umożliwia skryptom modyfikowanie elementów, treści, struktury oraz stylu dokumentu (HTML, CSS)

Hierarchia obiektów JavaScript

navigator

dostarcza informacje o przeglądarce, takie jak wersja, nazwa i aktywne pluginy.

userAgent: Zwraca string reprezentujący agenta użytkownika.

language: Zwraca preferowany język użytkownika.

```
console.log('Przeglądarka:', navigator.userAgent);  
console.log('Preferowany język:', navigator.language);
```

Array

globalny konstruktor służący do tworzenia i zarządzania listami i kolekcjami danych.

length: Właściwość określająca liczbę elementów w tablicy.

push() / pop(): Metody do dodawania elementów na końcu tablicy i usuwania ostatniego elementu.

```
let liczby = [3, 1, 4, 1, 5];  
console.log(liczby.length); // Wynik: 5  
  
liczby.push(9); // Dodaje 9 na końcu tablicy  
console.log(liczby); // Wynik: [3, 1, 4, 1, 5, 9]  
  
let ostatni = liczby.pop(); // Usuwa ostatni element, zwraca 9  
console.log(ostatni); // Wynik: 9
```

Hierarchia obiektów JavaScript

Date

reprezentuje datę i czas. Umożliwia tworzenie, interpretację i manipulację datami oraz czasem, oferując metody do zarządzania datami w różnych formatach.

getDate() / setDate(): Pobiera lub ustawia dzień miesiąca (1-31).

getDay(): Pobiera dzień tygodnia (0-6) dla określonej daty.

getMonth() / setMonth(): Pobiera lub ustawia miesiąc (0-11).

getFullYear() / setFullYear(): Pobiera lub ustawia rok.

getHours() / setHours(): Pobiera lub ustawia godzinę dnia (0-23).

getTime() / setTime(): Pobiera lub ustawia czas w milisekundach od 1 stycznia 1970 roku.

toString(): Zwraca string reprezentujący określoną datę.

now(): Metoda zwracająca aktualny czas w milisekundach..

```
let teraz = new Date();
console.log('Dzisiaj jest:', teraz.toString()); // Wyświetla aktualną datę i czas

let urodziny = new Date('1990-05-15');
console.log('Dzień tygodnia urodzin:', urodziny.getDay()); // Wynik: Dzień tygodnia

urodziny.setFullYear(1991);
console.log('Nowy rok urodzin:', urodziny.getFullYear()); // Ustawia rok na 1991

let milisekundy = Date.now();
console.log('Czas w milisekundach od 1 stycznia 1970 r.:', milisekundy);
```

Hierarchia obiektów JavaScript

Math

posiada właściwości i metody do wykonywania operacji matematycznych. Nie jest konstruktorem i wszystkie jego właściwości oraz metody są statyczne.

Math.PI: Stała reprezentująca wartość liczby Pi.

Math.round(): Zaokrągla liczbę do najbliższej liczby całkowitej.

Math.ceil(): Zaokrągla liczbę w górę do najbliższej liczby całkowitej.

Math.floor(): Zaokrągla liczbę w dół do najbliższej liczby całkowitej.

Math.random(): Zwraca pseudolosową liczbę między 0 a 1.

Math.sqrt(): Oblicza pierwiastek kwadratowy z liczby.

Math.pow(): Podnosi liczbę do określonej potęgi.

Math.abs(): Zwraca wartość absolutną (moduł) liczby.

```
console.log('Wartość Pi:', Math.PI);
console.log('Zaokrąglenie 3.5:', Math.round(3.5)); // Wynik: 4

let losowa = Math.random();
console.log('Liczba losowa:', losowa);

let pierwiastek = Math.sqrt(16);
console.log('Pierwiastek kwadratowy z 16:', pierwiastek); // Wynik: 4

let potega = Math.pow(2, 3);
console.log('2 do potęgi 3:', potega); // Wynik: 8

let wartoscBezwzględna = Math.abs(-10);
console.log('Wartość bezwzględna -10:', wartoscBezwzględna); // Wynik: 10
```

Hierarchia obiektów JavaScript

RegExp

reprezentuje wyrażenia regularne, które są wzorcami służącymi do dopasowywania kombinacji znaków w ciągach tekstowych. Wykorzystywane są one do wyszukiwania, walidacji, i manipulacji tekstem..

test(): Testuje, czy wzorzec występuje w podanym ciągu i zwraca true lub false.

exec(): Wykonuje wyszukiwanie dopasowania w ciągu i zwraca wynik w postaci tablicy lub null.

flags: Zwraca ciąg znaków oznaczających flagi wyrażenia regularnego.

source: Zwraca tekst wzorca wyrażenia regularnego.

```
let wzorzec = /ab+c/i;
console.log('Czy wzorzec pasuje do "Abc"?', wzorzec.test('Abc')); // Wynik: true

let wynik = wzorzec.exec('Pamiętaj o abcd i ABC');
console.log('Wynik dopasowania:', wynik[0]); // Wynik: "Abc"

console.log('Flagi wzorca:', wzorzec.flags); // Wynik: "i"
console.log('Tekst wzorca:', wzorzec.source); // Wynik: "ab+c"

let regexGlobalny = /abc/g;
console.log('Czy wzorzec globalny?', regexGlobalny.global); // Wynik: true
```

Hierarchia obiektów JavaScript

Symbol/Wzorzec	Opis
.	Dopasowuje dowolny pojedynczy znak, z wyjątkiem znaku nowej linii.
^	Dopasowuje początek ciągu znaków lub linii.
\$	Dopasowuje koniec ciągu znaków lub linii.
*	Dopasowuje zero lub więcej wystąpień poprzedzającego elementu.
+	Dopasowuje jedno lub więcej wystąpień poprzedzającego elementu.
?	Dopasowuje zero lub jedno wystąpienie poprzedzającego elementu.
{n}	Dopasowuje dokładnie n wystąpień poprzedzającego elementu.
{n,}	Dopasowuje n lub więcej wystąpień poprzedzającego elementu.
{n,m}	Dopasowuje od n do m wystąpień poprzedzającego elementu.
[abc]	Dopasowuje dowolny pojedynczy znak z podanych (w tym przypadku a, b lub c).
[^abc]	Dopasowuje dowolny pojedynczy znak, który nie jest wymieniony (w tym przypadku nie a, b ani c).
(abc)	Dopasowuje dokładnie sekwencję abc (grupowanie).
\s	Dopasowuje dowolny biały znak, w tym spację, tabulację, znak nowej linii.
\S	Dopasowuje dowolny znak inny niż biały znak.
\d	Dopasowuje dowolną cyfrę (0-9).
\D	Dopasowuje dowolny znak, który nie jest cyfrą.
\w	Dopasowuje dowolny znak alfanumeryczny, w tym cyfrę lub podkreślenie (_).
\W	Dopasowuje dowolny znak, który nie jest znakiem alfanumerycznym ani podkreśleniem.
\b	Dopasowuje granicę słowa, czyli pozycję pomiędzy znakiem \w a \W (lub na początku/koniec ciągu).
\B	Dopasowuje pozycję, która nie jest granicą słowa.
\\	Używany do escapowania znaków specjalnych, np. \\ dopasowuje literał kropki.

```
const emailRegex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;  
const testEmail1 = "przykladowy.email@example.com";  
const testEmail2 = "nieprawidlowyemail@przyklad";
```

```
console.log(emailRegex.test(testEmail1)); // Wynik: true  
console.log(emailRegex.test(testEmail2)); // Wynik: false
```

```
const kodPocztowyRegex = /^[0-9]{2}-[0-9]{3}$/;  
const testKodPocztowy1 = "00-123";  
const testKodPocztowy2 = "00123";
```

```
console.log(kodPocztowyRegex.test(testKodPocztowy1)); // Wynik: true  
console.log(kodPocztowyRegex.test(testKodPocztowy2)); // Wynik: false
```

```
const peselRegex = /^[0-9]{11}$/;  
const testPesel1 = "85050515836";  
const testPesel2 = "1234567890A";
```

```
console.log(peselRegex.test(testPesel1)); // Wynik: true  
console.log(peselRegex.test(testPesel2)); // Wynik: false
```

DOM (Document Object Model)

window

reprezentuje okno przeglądarki. Działa jako globalny obiekt dla skryptów na stronie, co oznacza, że wszystkie globalne zmienne i funkcje są w rzeczywistości właściwościami tego obiektu.

document: Właściwość odnosząca się do obiektu document, który reprezentuje załadowany dokument HTML.

location: Zawiera informacje o bieżącym URL i pozwala na przekierowywanie do innych adresów.

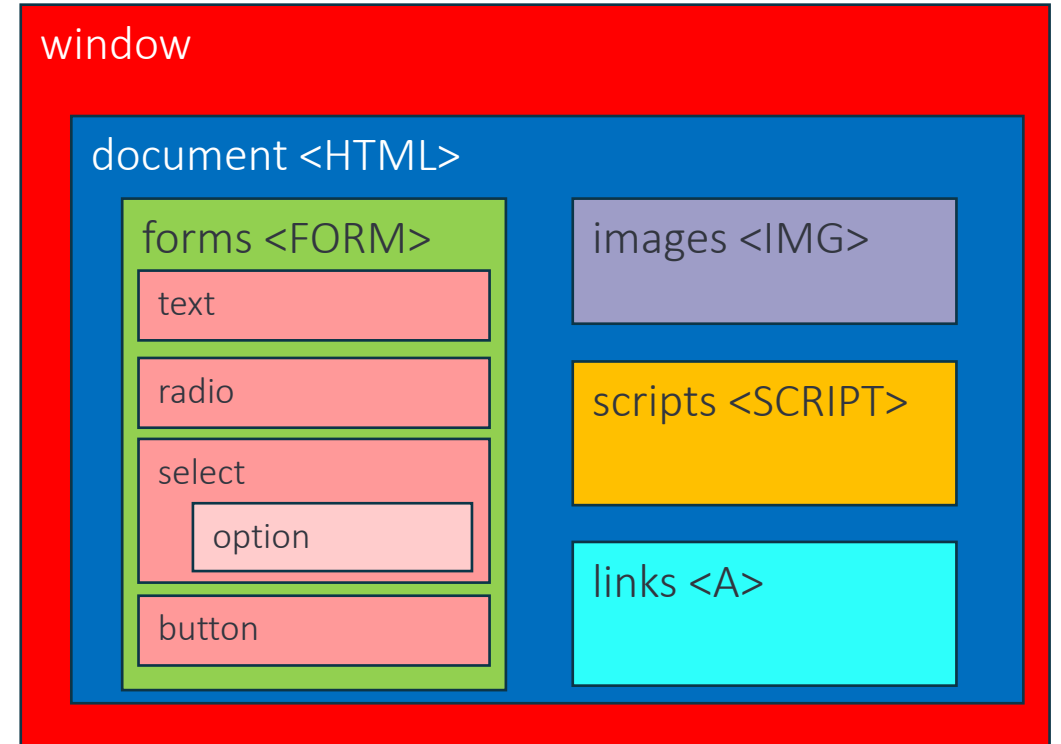
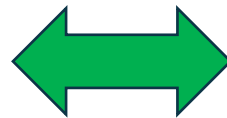
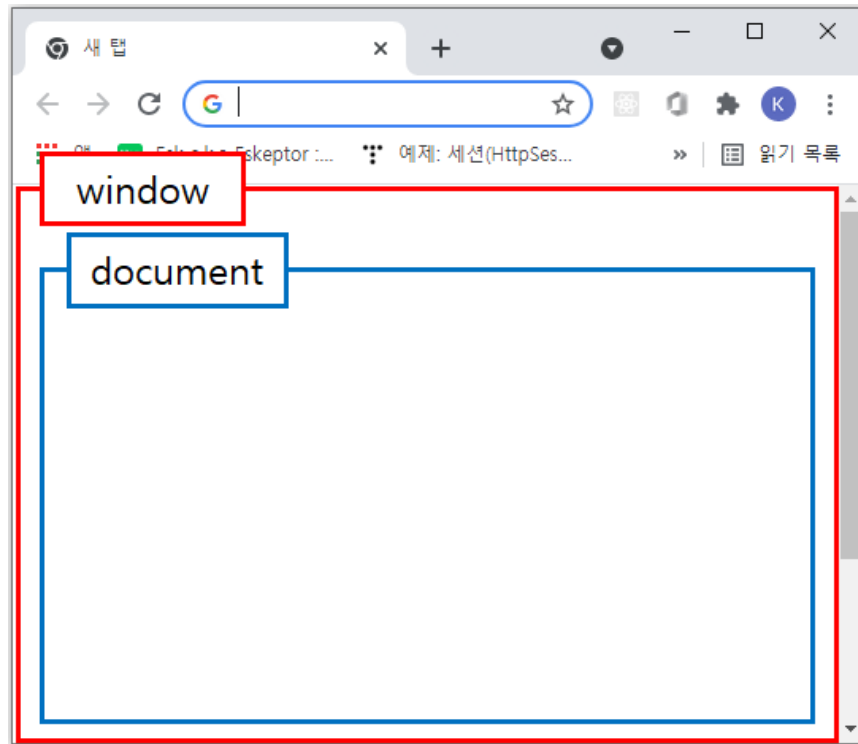
history: Umożliwia interakcję z historią przeglądania.

innerWidth / innerHeight: Właściwości określające wymiary wewnętrzne okna przeglądarki.

```
let szerokosc = window.innerWidth;  
let wysokosc = window.innerHeight;  
console.log(`Wymiary okna: ${szerokosc}x${wysokosc}px`);
```

```
// Wyświetlenie bieżącego URL  
console.log('Aktualny URL:', window.location.href);
```

DOM (Document Object Model)



DOM (Document Object Model)

window

setTimeout() / clearTimeout(): Pozwalają na uruchomienie funkcji po określonym czasie i jej anulowanie.

1/1000 sekundy

setInterval() / clearInterval(): Pozwalają na cykliczne wykonywanie funkcji w określonych odstępach czasowych i ich anulowanie.

```
// setTimeout: Uruchamia funkcję po upływie określonego czasu (np. 2 sekundy).  
const timeoutId = setTimeout(() => {  
  console.log('To wiadomość wyświetli się po 2 sekundach.');}, 2000);
```

```
// setInterval: Cyklicznie uruchamia funkcję co określony czas (np. co 3 sekundy).  
const intervalId = setInterval(() => {  
  console.log('Ta wiadomość będzie się wyświetlać co 3 sekundy.');}, 3000);
```

DOM (Document Object Model)

window

alert(): Wyświetla okno dialogowe z wiadomością i oczekuje, aż użytkownik naciśnie "OK".

confirm(): Wyświetla okno dialogowe z pytaniem (w formie wartości typu boolean) i dwoma przyciskami: "OK" i "Anuluj".

prompt(): Wyświetla okno dialogowe z polem tekstowym dla użytkownika i dwoma przyciskami: "OK" i "Anuluj".

```
alert('To jest komunikat alertu.');
```

 -> nic nie zwraca

```
const isConfirmed = confirm('Czy na pewno chcesz to zrobić?');  
console.log(isConfirmed); // Zwróci true, jeśli użytkownik kliknie
```

-> zwraca true lub false

```
const userInput = prompt('Wprowadź swoje imię:');  
console.log(userInput); // Zwróci wartość wprowadzoną przez użytkownika
```

-> zwraca wprowadzoną wartość przez użytkownika

DOM (Document Object Model)

document (window.document)

to główny punkt wejścia do całej zawartości strony internetowej i zapewnia interfejs do drzewa DOM. Pozwala na dynamiczne dostęp do i modyfikacje struktury, stylu oraz zawartości dokumentu.

getElementById(): Zwraca pojedynczy element o określonym ID

getElementsByClassName(): Zwraca listę elementów o określonej nazwie klasy.

getElementsByTagName(): Zwraca listę elementów o określonym tagu.

body: Zwraca element <body> dokumentu HTML.

head: Zwraca element <head> dokumentu HTML.

title: Umożliwia odczytanie lub ustawienie tytułu dokumentu.

URL: Zwraca pełny URL dokumentu.

referrer: Zwraca adres URL dokumentu, który przekierował do bieżącej strony.

cookie: Umożliwia odczytanie lub ustawienie ciasteczek (cookies) dla dokumentu.

forms: Zwraca kolekcję formularzy znajdujących się w dokumencie.

links: Zwraca kolekcję wszystkich hiperłączy w dokumencie.

images: Zwraca kolekcję wszystkich obrazów w dokumencie.

scripts: Zwraca kolekcję wszystkich skryptów w dokumencie.

styleSheets: Zwraca kolekcję arkuszy stylów powiązanych z dokumentem.

DOM (Document Object Model)

ID vs. Class (CSS)

ID jest unikalne. Na całej stronie HTML będzie tylko jeden atrybut ID z unikalną nazwą.

Class (klasa) nie jest unikalna. Możemy używać tej samej nazwy klasy w wielu elementach oraz kilka klas w tym samym elemencie.

ID	class (klasa)
Przypisanie atrybutu id do znacznika HTML, aby zastosować stylizację lub interaktywność do konkretnego znacznika HTML	Przypisanie atrybutu class do liczby znaczników HTML, aby zastosować stylizację lub interaktywność do grupy znaczników HTML
Selektor ID używa znaku #	Selektor klasy używa znaku "."
Składnia to #id{ stylowanie CSS;}	Składnia to .class{ stylowanie CSS;}

DOM (Document Object Model)

```
<style>
  #unikalny-identyfikator {
    color: blue;
  }

  .grupa-elementow {
    font-size: 16px;
  }
</style>
```

← ID

← class

```
<div id="unikalny-identyfikator">Tekst z unikalnym identyfikatorem</div>
<div class="grupa-elementow">Pierwszy element grupy</div>
<div class="grupa-elementow">Drugi element grupy</div>
<p class="grupa-elementow">Trzeci element grupy w paragrafie</p>
```

```
<script>
  // Używanie getElementById
  const unikalnyElement = document.getElementById('unikalny-identyfikator');
  console.log('Element z ID:', unikalnyElement);
  unikalnyElement.style.backgroundColor = 'yellow';

  // Używanie getElementsByClassName
  const grupaElementow = document.getElementsByClassName('grupa-elementow');
  console.log('Elementy z klasą:', grupaElementow);
  Array.from(grupaElementow).forEach(elem => {
    elem.style.color = 'red';
  });

  // Używanie getElementsByTagName
  const wszystkieDivy = document.getElementsByTagName('div');
  console.log('Wszystkie elementy <div>:', wszystkieDivy);
  Array.from(wszystkieDivy).forEach(div => {
    div.style.border = '1px solid black';
  });
</script>
```

DOM (Document Object Model)

document.body

reprezentuje element <body> w dokumencie HTML, który jest kontenerem na wszystkie elementy tworzące zawartość strony.

style: Obiekt, który pozwala na modyfikację stylów CSS dla elementu <body>.

getAttribute(): Pobiera wartość atrybutu elementu <body>.

setAttribute(): Ustawia wartość atrybutu elementu <body>.

innerHTML: Zawartość HTML wewnątrz elementu <body>.

```
// Modyfikacja stylów CSS dla <body>
document.body.style.backgroundColor = 'lightblue';
document.body.style.margin = '0';
```

```
// Pobieranie wartości atrybutu (np. 'class') elementu <body>
console.log(document.body.getAttribute('class'));

// Ustawianie wartości atrybutu elementu <body> (np. dodanie klasy 'active')
document.body.setAttribute('class', 'active');
```

```
// Pobieranie zawartości HTML wewnątrz <body>
console.log(document.body.innerHTML);

// Ustawianie nowej zawartości HTML wewnątrz <body>
document.body.innerHTML = '<h1>Witaj, świecie!</h1><p>To jest nowy akapit.</p>';
```

DOM (Document Object Model)

Właściwość CSS	Notacja JS	Przykład użycia (CSS)	Przykład użycia w JS
color	color	color: red;	element.style.color = 'red';
background-color	backgroundColor	background-color: #f0f0f0;	element.style.backgroundColor = '#f0f0f0';
border	border	border: 1px solid black;	element.style.border = '1px solid black';
padding	padding	padding: 10px;	element.style.padding = '10px';
margin	margin	margin: 20px;	element.style.margin = '20px';
display	display	display: none;	element.style.display = 'none';
width	width	width: 100px;	element.style.width = '100px';
height	height	height: 200px;	element.style.height = '200px';
font-size	fontSize	font-size: 16px;	element.style.fontSize = '16px';
text-align	textAlign	text-align: center;	element.style.textAlign = 'center';
position	position	position: absolute;	element.style.position = 'absolute';
top	top	top: 50px;	element.style.top = '50px';
opacity	opacity	opacity: 0.5;	element.style.opacity = '0.5';
z-index	zIndex	z-index: 10;	element.style.zIndex = '10';
overflow	overflow	overflow: hidden;	element.style.overflow = 'hidden';
font-weight	fontWeight	font-weight: bold;	element.style.fontWeight = 'bold';
font-style	fontStyle	font-style: italic;	element.style.fontStyle = 'italic';
text-decoration	textDecoration	text-decoration: underline;	element.style.textDecoration = 'underline';

DOM (Document Object Model)

document.body

zdarzenia to akcje lub wystąpienia, które mogą zostać wykryte przez skrypt, pozwalając na wykonanie kodu, gdy zdarzenia te zachodzą.

onload: Zdarzenie wyzwalane, gdy cały dokument został załadowany, w tym element <body>.

onresize: Zdarzenie wyzwalane, gdy zmienia się rozmiar okna przeglądarki.

onscroll: Zdarzenie wyzwalane, gdy użytkownik przewija element <body>.

```
<body onload="alert('Dokument został załadowany!');"
      onresize="alert('Zmieniono rozmiar okna przeglądarki!');"
      onscroll="alert('Przewijanie elementu body!');">
```

```
<script>
  // Reaguje na załadowanie całego dokumentu, w tym elementu <body>
  window.onload = function() {
    console.log('Dokument został załadowany!');
  };

  // Reaguje na zmianę rozmiaru okna przeglądarki
  window.onresize = function() {
    console.log('Zmieniono rozmiar okna przeglądarki!');
  };

  // Reaguje na przewijanie treści na stronie
  window.onscroll = function() {
    console.log('Przewijanie elementu body!');
  };
</script>
```

DOM (Document Object Model)

document.head

właściwość zwraca odniesienie do elementu <head> w dokumencie HTML. Jest to użyteczne, gdy potrzebujemy dostępu do metadanych strony, takich jak tytuł, linki do arkuszy stylów czy skrypty.

```
// Pobieranie elementu <head>
const headElement = document.head;
console.log(headElement);
```

document.title

właściwość umożliwia odczytanie bieżącego tytułu dokumentu lub jego ustawienie. Standardowo tytuł dokumentu jest wyświetlany na pasku tytułowym przeglądarki lub na karcie strony.

```
// Odczytanie tytułu dokumentu
console.log('Aktualny tytuł strony:', document.title);

// Ustawienie nowego tytułu dokumentu
document.title = 'Nowy tytuł strony';
```

Te właściwości są szczególnie przydatne w scenariuszach, kiedy trzeba zaktualizować metadane strony po jej załadowaniu lub podczas dynamicznego manipulowania zawartością dokumentu, np. w aplikacjach jednostronicowych (SPA), gdzie zawartość strony zmienia się bez przeładowania całej strony.

DOM (Document Object Model)

document.URL

właściwość zapewnia pełny adres URL dokumentu, który jest aktualnie wyświetlany przez przeglądarkę. Zawiera protokół, host, port (jeśli określony), ścieżkę dostępu, parametry zapytania oraz fragment identyfikatora.

```
// Wyświetlenie pełnego URL dokumentu  
console.log('Pełny URL dokumentu:', document.URL);
```

document.referrer

właściwość zwraca adres URL strony, z której użytkownik został przekierowany do bieżącej strony. Jest to przydatne na przykład w analizie ruchu na stronie, aby zbudować log stron, skąd przychodzą użytkownicy.

```
// Wyświetlenie adresu URL strony, z której użytkownik został przekierowany  
console.log('URL strony referencyjnej:', document.referrer);
```

DOM (Document Object Model)

document.cookie

Właściwość ta pozwala na odczytanie oraz zapis ciasteczek (cookies) dla bieżącego dokumentu.

Ciasteczka są wykorzystywane do przechowywania danych sesji użytkownika, jak preferencje czy status logowania, między różnymi żądaniami strony.

Większość przeglądarek ogranicza rozmiar pojedynczego ciasteczka do około 4096 bajtów (4 KB).

```
// Zapisanie ciasteczka
document.cookie = 'userSettings=darkMode; expires=Fri, 31 Dec 9999 23:59:59 GMT; path=/';

// Odczytanie wszystkich ciasteczek
console.log('Ciasteczka dostępne dla dokumentu:', document.cookie);

// Usunięcie ciasteczka poprzez ustawienie daty wygaśnięcia w przeszłości
document.cookie = 'userSettings=; expires=Thu, 01 Jan 1970 00:00:00 GMT; path=/';
```

DOM (Document Object Model)

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 zapytaniami przesyłanymi przez HTTPS.
HttpOnly	Kiedy ten atrybut jest ustawiony, ciasteczko nie jest dostępne przez skrypty klienta (np. JavaScript), co pomaga zapobiegać atakom typu XSS.

