

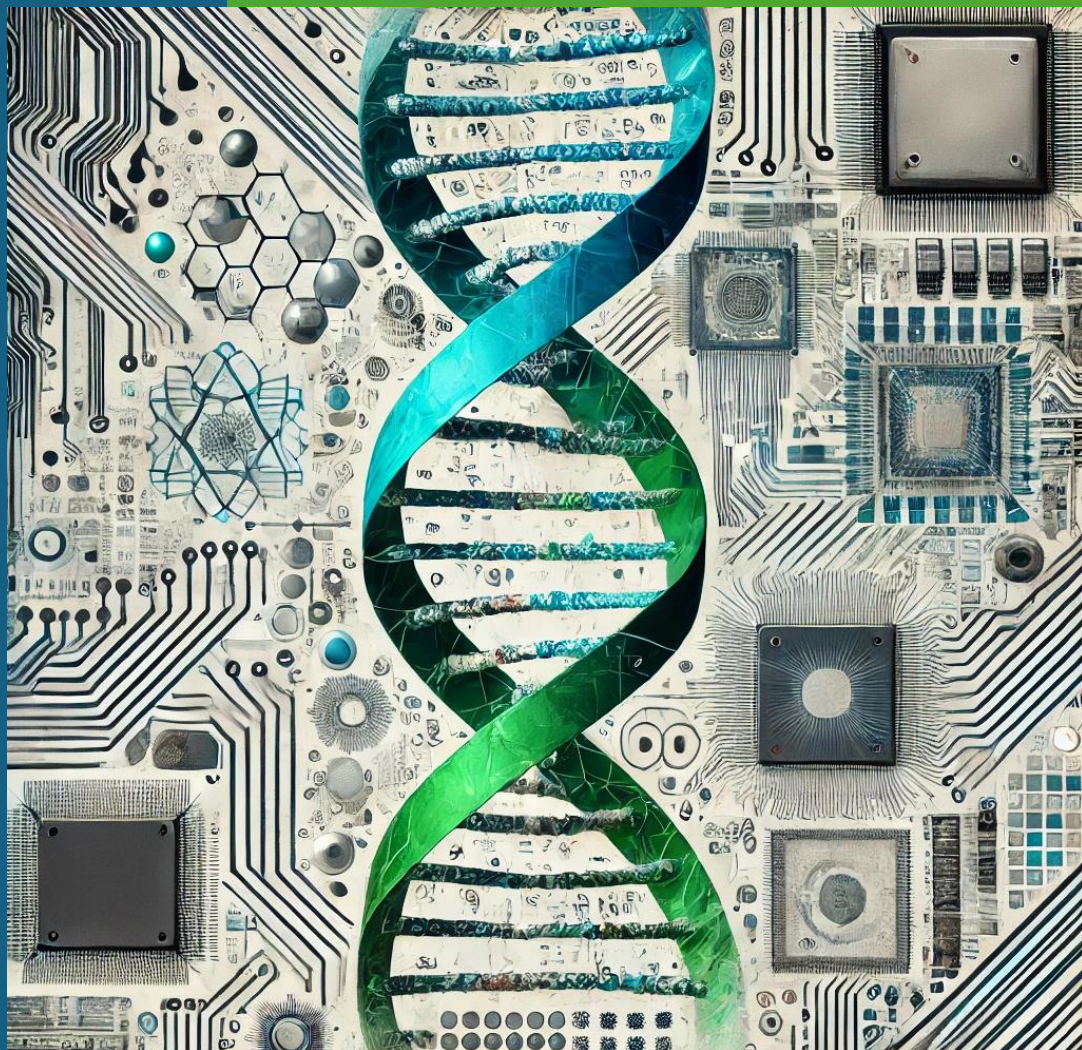
Temat 2 .

Architektura dokumentów WWW

dr inż. Michał Malinowski

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

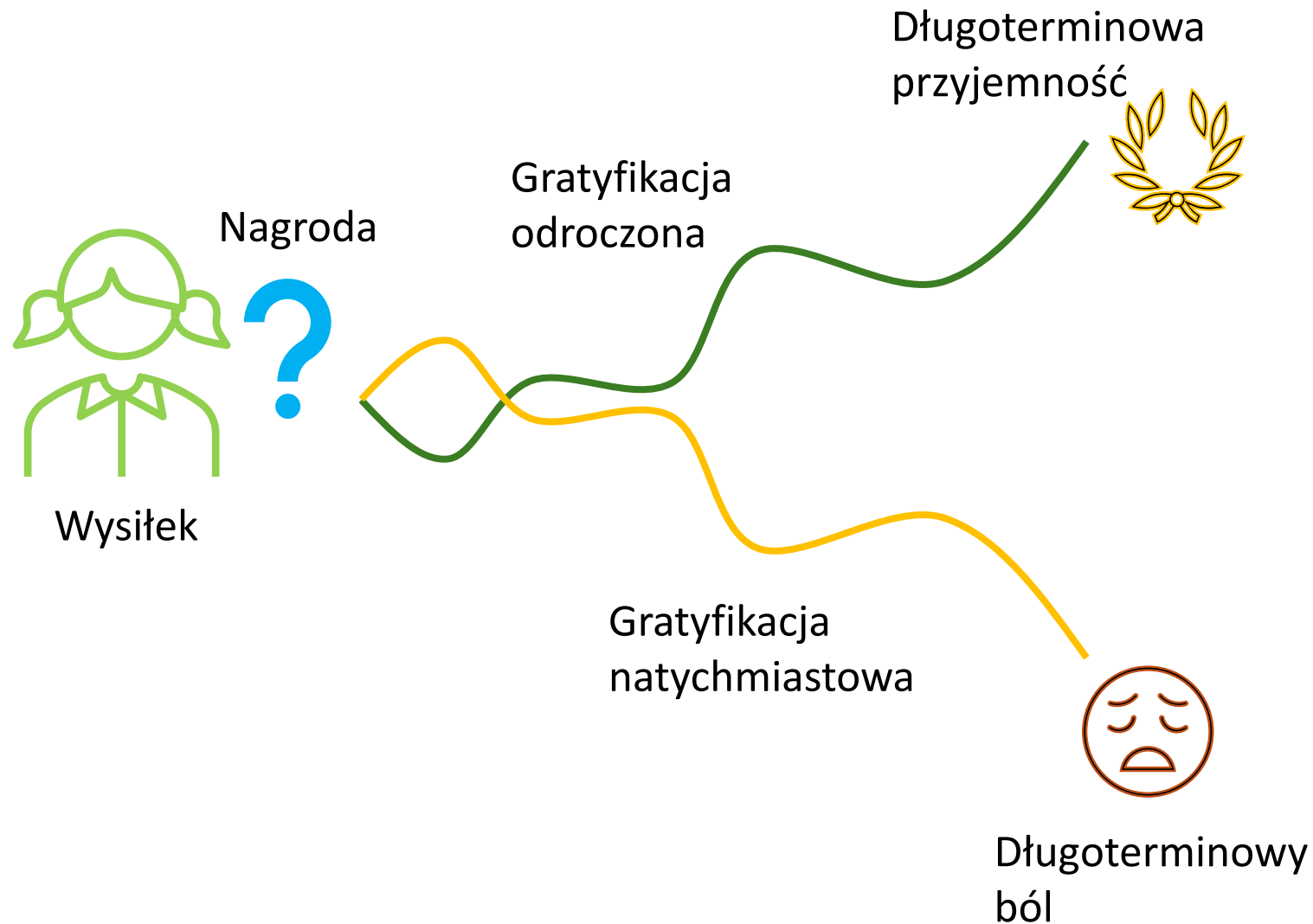
TECHNIKI INTERNETOWE



Zagadnienia:

- HTML
- CSS
- JavaScript
- JSON
- DOM

Gratyfikacja natychmiastowa vs odroczone



Architektura dokumentów WWW

Architektura dokumentów WWW

odnosi się do struktury i organizacji dokumentów na stronach internetowych, która umożliwia efektywne zarządzanie, prezentację i dostęp do treści w sieci. Poniżej przedstawiono główne elementy tej architektury:

Elementy Architektury

HTML (HyperText Markup Language):

- Język znaczników używany do tworzenia i strukturyzowania treści na stronach internetowych.

CSS (Cascading Style Sheets):

- Język służący do opisu wyglądu i formatowania dokumentu HTML.

JavaScript:

- Język skryptowy używany do tworzenia interaktywności na stronach internetowych.

HTTP/HTTPS (HyperText Transfer Protocol / Secure):

- Protokół komunikacyjny używany do przesyłania danych między przeglądarką a serwerem WWW.

Serwer WWW:

- Oprogramowanie lub urządzenie, które przetwarza żądania HTTP/HTTPS i zwraca odpowiednie zasoby.

DNS (Domain Name System):

- System zamieniający nazwy domen (np. `www.example.com`) na adresy IP.

Frameworki i Biblioteki:

- Narzędzia i zasoby wspomagające rozwój stron internetowych.

HTML

Język HTML (HyperText Markup Language)

to język deklaratywny znaczników używany do tworzenia i strukturyzowania dokumentów w WWW.

HTML wywodzi się z SGML, czyli Standard Generalized Markup Language.

1989: Tim Berners-Lee opracowuje HTML jako język do tworzenia dokumentów hipertekstowych.

1991: Publiczne udostępnienie HTML.

1997: W3C (World Wide Web Consortium) staje się opiekunem standardu HTML.

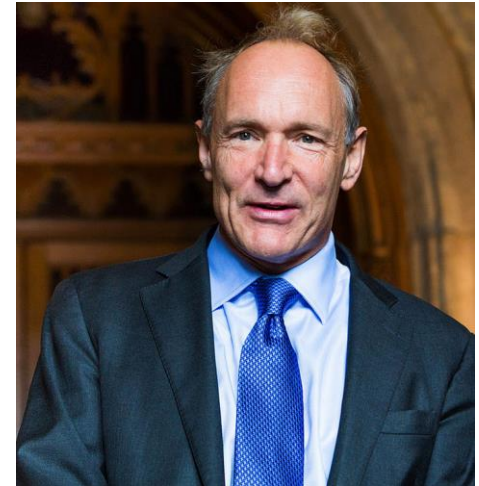
Wersja HTML	Rok Wydania
HTML 1.0	1993
HTML 2.0	1995
HTML 3.2	1997
HTML 4.0	1997
HTML 4.01	1999
XHTML 1.0	2000
HTML5	2014

HTML

Sir Timothy John Berners-Lee

(ur. 8 czerwca 1955 w Londynie) – brytyjski fizyk i programista, współtwórca i jeden z pionierów usługi WWW.

- **1989:** podczas pracy w CERN (Europejska Organizacja Badań Jądrowych) w Szwajcarii, Berners-Lee przedstawił propozycję systemu hipertekstowego, który umożliwiłby badaczom łatwe udostępnianie dokumentów.
- **1990:** stworzył pierwszą przeglądarkę internetową oraz pierwszy serwer WWW.
- **1991:** publicznie uruchomił World Wide Web, umożliwiając globalną wymianę
- **1994:** założył W3C, organizację zajmującą się opracowywaniem standardów dla sieci WWW, aby zapewnić jej długoterminowy rozwój i dostępność dla wszystkich.



HTML

Struktura dokumentu

- `<!DOCTYPE html>`: Deklaracja typu dokumentu.
- `<html>`: Element główny, obejmuje cały dokument HTML.
- `<head>`: Sekcja nagłówkowa zawierająca metadane, tytuł dokumentu, odnośniki do stylów i skryptów.
- `<body>`: Sekcja treściowa dokumentu zawierająca rzeczywiste treści wyświetlane użytkownikom.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Tytuł Strony</title>
</head>
<body>
  <h1>Witamy na naszej stronie!</h1>
  <p>To jest przykładowa strona HTML.</p>
</body>
</html>
```

HTML

Metatagi

to elementy HTML umieszczane w sekcji <head> dokumentu, które dostarczają przeglądarkom i innym usługom internetowym dodatkowych informacji o stronie.

Metatagi nie są widoczne dla użytkowników, ale są kluczowe dla SEO (optymalizacji pod kątem wyszukiwarek), analizy stron oraz dostosowania strony do różnych urządzeń i przeglądarek.

Informacyjne:

- Kodowanie znaków: `<meta charset="UTF-8">`
- Opis strony: `<meta name="description" content="Opis strony">`
- Słowa kluczowe: `<meta name="keywords" content="HTML, CSS, JavaScript">`
- Autor strony: `<meta name="author" content="Jan Kowalski">`

HTML

Metatagi

Kontrolujące Wyświetlanie:

- Responsywność: `<meta name="viewport" content="width=device-width, initial-scale=1.0">`
- Tryb renderowania: `<meta http-equiv="X-UA-Compatible" content="IE=edge">`

SEO i Social Media:

- Tytuł: `<meta property="og:title" content="Tytuł strony">`
- Opis: `<meta property="og:description" content="Opis strony">`
- Obraz: `<meta property="og:image" content="URL do obrazu">`

```
<head>
  <meta charset="UTF-8">
  <title>Przykładowa Strona</title>
  <meta name="description" content="To jest przykładowy opis strony.">
  <meta name="keywords" content="HTML, CSS, JavaScript, metatagi">
  <meta name="author" content="Jan Kowalski">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta property="og:title" content="Przykładowy tytuł strony">
  <meta property="og:description" content="Przykładowy opis strony">
  <meta property="og:image" content="https://www.example.com/image.jpg">
</head>
```

HTML

Główne Tagi HTML

to składnik który służy do definiowania struktury i zawartości dokumentu internetowego.

- Nagłówki: `<h1>`, `<h2>`, `<h3>`, ..., `<h6>`
- Akapity: `<p>`
- Linki: `<a href="URL">`
- Obrazy: ``
- Listy:
 - Uporządkowane: `<ol>`, `<li>`
 - Nieuporządkowane: `<ul>`, `<li>`

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Moja Strona</title>
</head>
<body>
  <h1>Witamy na mojej stronie!</h1>
  <p>To jest przykładowa strona HTML.</p>
  <a href="https://www.example.com">Odwiedź naszą stronę</a>
  <ul>
    <li>Punkt 1</li>
    <li>Punkt 2</li>
    <li>Punkt 3</li>
  </ul>
</body>
</html>
```

HTML

Atrybuty HTML

Atrybuty dostarczają dodatkowe informacje o elementach HTML. Są umieszczane w otwierającym znaczniku elementu.

- *href*: Określa URL w znaczniku `<a>`
- *src*: Określa źródło obrazu w `<img>`
- *alt*: Dostarcza alternatywny tekst dla obrazów w `<img>`
- *id*: Unikalny identyfikator elementu.
- *class*: Klasa elementu, używana do stylizacji CSS

```
<a href="https://www.example.com" id="link1" class="nav-link">Odwiedź naszą stronę</a>  

```

HTML

Formaty Tekstowe i Znaki Specjalne

Formatowanie Tekstu

- Pogrubienie: `<b>`, `<strong>`
- Kursywa: `<i>`, `<em>`
- Podkreślenie: `<u>`

Znaki Specjalne

- Ampersand (&): `&`
- Mniejszy niż: `<`
- Większy niż: `>`
- Cudzysłów: `"`
- Apostrof: `'`
- Spacja: ` `

```
<h1>Formatowanie Tekstu</h1>
<ul>
  <li>Pogrubienie: <b>b</b>, <strong>strong</strong></li>
  <li>Kursywa: <i>i</i>, <em>em</em></li>
  <li>Podkreślenie: <u>u</u></li>
</ul>
<h1>Znaki Specjalne</h1>
<ul>
  <li>Ampersand: &amp;</li>
  <li>Mniejszy niż: &lt;</li>
  <li>Większy niż: &gt;</li>
  <li>Cudzysłów: &quot;</li>
  <li>Apostrof: &apos;</li>
  <li>Spacja: &nbsp;&nbsp;&nbsp;&nbsp;spacje</li>
</ul>
```

HTML

Formularze HTML

to interaktywne elementy na stronach internetowych, które pozwalają użytkownikom na wprowadzanie danych i przesyłanie ich do serwera.

- Formularz: `<form>` (atrybuty: `action`, `method`).
- Pola tekstowe: `<input type="text">`
- Pola hasła: `<input type="password">`
- Przyciski: `<button>`, `<input type="submit">`
- Listy rozwijane: `<select>`, `<option>`

```
<form action="/submit" method="post">
  <label for="name">Imię:</label>
  <input type="text" id="name" name="name">
  <label for="password">Hasło:</label>
  <input type="password" id="password" name="password">
  <button type="submit">Wyślij</button>
</form>
```

HTML

Nowe Elementy HTML5

- Strukturalne: `<header>`, `<nav>`, `<section>`, `<article>`, `<footer>`
- Multimedialne: `<audio>`, `<video>`
- Wprowadzania danych: `<input type="email">`, `<input type="date">`

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Przykład HTML5</title>
</head>
<body>
  <header><h1>Moja Strona</h1></header>
  <nav><a href="#home">Home</a> | <a href="#about">About</a></nav>
  <section>
    <article><h2>Artykuł</h2><p>Treść artykułu.</p></article>
  </section>
  <footer><p>&copy; 2023 Moja Strona</p></footer>
  <audio controls><source src="audio.mp3" type="audio/mpeg">Audio not supported.</audio>
  <video controls width="320"><source src="video.mp4" type="video/mp4">Video not supported.</video>
  <form>
    <label>Email: <input type="email" required></label><br>
    <label>Data: <input type="date" required></label><br>
    <button type="submit">Wyślij</button>
  </form>
</body>
</html>
```

HTML

Kodowań Stron Internetowych

```
<meta charset="UTF-8">
```

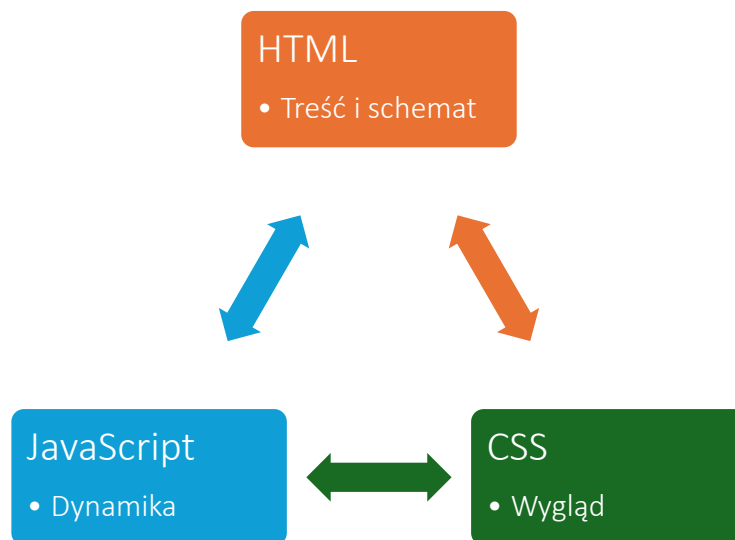
Kodowanie	Opis	Zalety	Wady
UTF-8	Uniwersalne kodowanie obsługujące wszystkie znaki Unicode	- Uniwersalne - Kompatybilne z większością języków - Oszczędne pod względem miejsca	- Może zajmować więcej miejsca dla znaków ASCII
ISO-8859-1 (Latin-1)	Kodowanie obsługujące zachodnioeuropejskie języki	- Szeroko stosowane w przeszłości - Proste do implementacji	- Ograniczone w porównaniu do UTF-8 - Nie obsługuje wielu znaków spoza zachodnioeuropejskich języków
UTF-16	Kodowanie obsługujące wszystkie znaki Unicode, używa 16-bitowych jednostek kodowych	- Obsługuje wszystkie znaki Unicode - Efektywne dla języków azjatyckich	- Może zajmować więcej miejsca dla znaków ASCII - Mniej powszechne niż UTF-8
ASCII	Podstawowe kodowanie, obsługujące 128 znaków	- Bardzo oszczędne - Proste i szeroko zrozumiane	- Ograniczone do 128 znaków - Nie obsługuje znaków międzynarodowych
Windows-1250	Kodowanie obsługujące języki środkowoeuropejskie, takie jak polski, czeski, węgierski	- Dobrze obsługuje języki środkowoeuropejskie - Proste do implementacji	- Ograniczone w porównaniu do UTF-8 - Mniej uniwersalne niż UTF-8

CSS

CSS – Kaskadowe Arkusze styli (Cascading Style Sheets)

to język deklaratywny służący do opisu wyglądu i formatowania dokumentów HTML.

Oddziela treść (HTML) od prezentacji (CSS), co ułatwia zarządzanie stronami internetowymi.



CSS

Selektory CSS

Określają, które elementy HTML mają być stylizowane.

- Selektory elementów (tagów): `p { }`
- Selektory klas: `.class { }`
- Selektory identyfikatorów: `#id { }`
- Selektory atrybutów: `[type="text"] { }`

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p { color: blue; }
    .class { font-size: 20px; }
    #id { background-color: yellow; }
    [type="text"] { border: 1px solid black; }
  </style>
</head>
<body>
  <p>To jest paragraf.</p>
  <p class="class">To jest paragraf z klasą.</p>
  <p id="id">To jest paragraf z identyfikatorem.</p>
  <input type="text" value="To jest pole tekstowe">
</body>
</html>
```

CSS

Właściwości CSS

są atrybutami używanymi do definiowania wyglądu i stylu elementów HTML na stronie internetowej.
Każda właściwość CSS składa się z nazwy (property) i wartości (value), które są przypisywane do selektorów.

Kategoria	Właściwość	Opis	Przykład
Kolory i Tło	color	Ustawia kolor tekstu	color: blue;
	background-color	Ustawia kolor tła	background-color: yellow;
Tekst	font-size	Ustawia rozmiar czcionki	font-size: 16px;
	font-family	Ustawia rodzinę czcionki	font-family: Arial, sans-serif;
	text-align	Ustawia wyrównanie tekstu	text-align: center;
	font-weight	Ustawia grubość czcionki	font-weight: bold;
Boksy	margin	Ustawia margines zewnętrzny	margin: 10px;
	padding	Ustawia margines wewnętrzny	padding: 20px;
	border	Ustawia obramowanie	border: 1px solid black;
Układ	display	Ustawia sposób wyświetlania elementu	display: flex;
	position	Ustawia pozycję elementu	position: absolute;
	width	Ustawia szerokość elementu	width: 100%;

CSS

Metody Ustawiania Wartości CSS

Inline (w kodzie)

Stylizowanie bezpośrednio w znaczniku HTML.

```
<p style="color: blue; font-size: 16px;">To jest paragraf.</p>
```

Osadzone

Dodawanie stylów w sekcji <head> dokumentu HTML.

```
<style>
  p { color: green; }
</style>
```

Zewnętrzne

Oddzielny plik CSS, dołączany za pomocą <link>.

```
<link rel="stylesheet" type="text/css" href="style.css">
```

Dynamiczne z
JavaScript

Modyfikowanie stylów za pomocą JavaScript.

```
document.getElementById("elem").style.color = "purple";
```

CSS

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;}

JavaScript

JavaScript

To interpretowany, obiektowy język programowania, który jest jednym z podstawowych technologii stosowanych w tworzeniu stron internetowych, obok HTML i CSS.

JavaScript pozwala na implementowanie interaktywnych funkcji, manipulację elementami na stronie (obiektami) i asynchroniczną komunikację z serwerem.

Skrypty JavaScript są umieszczane w dokumentach HTML za pomocą znacznika `<script>`

```
<script>
  console.log('Witaj, świecie!');
</script>
```

Mogą być osadzone w dokumencie HTML lub załadowane z zewnętrznego pliku.

```
<script src="script.js"></script>
```

JavaScript

Elementy składni

- Zmienne są deklarowane za pomocą *var*, *let* lub *const*
- Typy danych: *number*, *string*, *boolean*, *object*, *undefined*, *null*
- Operatory:
 - Aritmetyczne: *+*, *-*, ***, */*, *%*
 - Porównania: *==*, *===*, *!=*, *!==*, *<*, *>*, *<=*, *>=*
 - Logiczne: *&&*, *||*, *!*
- Funkcje
 - Definiowanie funkcji: Używając słowa kluczowego *function*.
 - Wywoływanie funkcji: Nazywając ją i przekazując argumenty.

```
let liczba = 10;  
const tekst = 'Witaj';  
var prawda = true;
```

```
let suma = 5 + 3; // 8  
let identyczny = (5 === '5'); // false  
let lubOperator = (true || false); // true
```

```
function przywitaj(imie) {  
    return 'Witaj, ' + imie;  
}  
console.log(przywitaj('Jan'));
```

JavaScript

ECMAScript 2024 (ES15)

jest najnowszą wersją standardu ECMAScript. Został oficjalnie zatwierdzony i opublikowany przez Ecma International w czerwcu 2024 roku.

ECMAScript jest standardem, na którym opiera się JavaScript oraz inne języki skryptowe, takie jak JScript (Microsoft) i ActionScript (Adobe).



ECMA "European Computer Manufacturers Association"

to międzynarodowa, prywatna organizacja non-profit założona w 1961 roku, mająca na celu rozwijanie i publikowanie standardów w dziedzinie informatyki i telekomunikacji. Organizacja ta zmieniła swoją nazwę na "Ecma International" w 1994 roku, aby lepiej odzwierciedlić jej globalny zasięg i działalność.

Podstawowe operatory w języku JavaScript

Operator	Opis	Przykład Użycia	Wynik Przykładu
+	Dodawanie	5 + 3	8
-	Odejmowanie	10 - 5	5
*	Mnożenie	4 * 7	28
/	Dzielenie	20 / 4	5
%	Reszta z dzielenia (modulo)	7 % 2	1
++	Inkrementacja (zwiększenie o 1)	let x = 5; x++;	x staje się 6
--	Dekrementacja (zmniejszenie o 1)	let y = 10; y--;	y staje się 9
==	Równość (bez sprawdzania typu)	5 == '5'	true
===	Równość (ze sprawdzaniem typu)	5 === '5'	false
!=	Nierówność (bez sprawdzania typu)	5 != '5'	false
!==	Nierówność (ze sprawdzaniem typu)	5 !== '5'	true
>	Większe niż	5 > 3	true
<	Mniejsze niż	5 < 10	true
>=	Większe lub równe	5 >= 5	true
<=	Mniejsze lub równe	5 <= 10	true
&&	Operator logiczny AND	true && false	false
	Operator logiczny OR	true false	true
!	Operator logiczny NOT	!true	false
? :	Operator warunkowy (ternary)	5 > 10 ? 'tak' : 'nie'	'nie'

Funkcje

Funkcje w programowaniu

to zorganizowane bloki instrukcji, które wykonują określone zadania.

Służą one do strukturyzowania kodu, umożliwiając jego wielokrotne wykorzystanie i ułatwiając zarządzanie złożonymi programami.

Można je wywołać, używając ich nazwy, aby uruchomić zawarty w nich kod, który może opcjonalnie zwracać wynik operacji.

Deklaracja funkcji

```
function nazwaFunkcji(parametr) {  
    const wynik = parametr * parametr;  
    return wynik;  
}
```

Wywołanie funkcji

```
nazwaFunkcji(2); // Zwraca 4  
nazwaFunkcji(3); // Zwraca 9  
nazwaFunkcji(5); // Zwraca 25
```

Funkcje

Deklaracja funkcji wewnątrz pliku

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Proste Funkcje w JavaScript</title>
</head>
<body>
  <h1>Wyniki Działañ</h1>
  <script>
    function dodaj(a, b) { return a + b; }
    function mnoz(a, b) { return a * b; }
    function odejmij(a, b) { return a - b; }
    console.log('Dodaj 2 + 3:', dodaj(2, 3));
    console.log('Mnóż 4 * 5:', mnoz(4, 5));
    console.log('Odejmij 10 - 6:', odejmij(10, 6));
  </script>
</body>
</html>
```

Funkcje

Deklaracja funkcji w pliku zewnętrznym „bibliotece”

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Przykład Użycia Prostej Biblioteki</title>
</head>
<body>
  <h1>Wyniki Działania</h1>
  <script src="simpleLib.js"></script>
  <script>
    // Użycie funkcji z biblioteki simpleLib.js
    console.log('Dodaj 2 + 3:', dodaj(2, 3)); // 5
    console.log('Mnoż 4 * 5:', mnoz(4, 5)); // 20
    console.log('Odejmij 10 - 6:', odejmij(10, 6)); // 4
  </script>
</body>
</html>
```

simpleLib.js

```
// Funkcja do dodawania dwóch liczb
function dodaj(a, b) {
  return a + b;
}

// Funkcja do mnożenia dwóch liczb
function mnoz(a, b) {
  return a * b;
}

// Funkcja do odejmowania dwóch liczb
function odejmij(a, b) {
  return a - b;
}
```

Funkcje

Parametry (argumenty) funkcji

W ES6 i nowszych wersjach JavaScript można określić domyślne wartości dla parametrów funkcji. Jeśli funkcja zostanie wywołana bez argumentu dla danego parametru, zostanie użyta wartość domyślna:

```
function powitaj(imie = "Gościu") {  
  console.log("Witaj, " + imie + "!");  
}
```

W ES6 wprowadzono także rest parameters, które pozwalają funkcji przyjąć nieokreśloną liczbę argumentów jako tablicę:

```
function suma(...numery) {  
  return numery.reduce((a, b) => a + b, 0);  
}
```

Obiekt arguments, który jest dostępny wewnątrz funkcji. Pozwala on na dostęp do wszystkich argumentów przekazanych do funkcji, niezależnie od tego, czy zostały one zdefiniowane w deklaracji funkcji, czy nie.

```
function przykladowaFunkcja() {  
  for(let i = 0; i < arguments.length; i++) {  
    console.log(arguments[i]);  
  }  
}
```

Funkcje

Różnica między arguments a rest parameters (...)

Cecha	arguments	Rest Parameters (...)
Typ	Obiekt podobny do tablicy	Prawdziwa tablica
Dostęp do wartości	Tak, poprzez indeksowanie arguments[i]	Tak, jako tablica z argumentami funkcji
Metody tablicowe	Nie, arguments nie jest instancją Array i nie posiada wbudowanych metod tablicowych	Tak, obsługuje wszystkie metody tablicowe (map, filter, reduce, itd.)
Definicja	Nie wymaga specjalnej składni, automatycznie dostępny w funkcjach	Definiowany jawnie w deklaracji funkcji, np. function przyklad(...args)
Dostępność	Dostępny we wszystkich typach funkcji, oprócz funkcji strzałkowych	Dostępny we wszystkich typach funkcji, włączając funkcje strzałkowe
ES6+	Dostępny, ale użycie rest parameters jest zalecane dla lepszej czytelności i funkcjonalności	Wprowadzony w ES6, oferuje nowoczesne i elastyczne podejście do obsługi argumentów

Funkcje

Zwracane wartości

Funkcje w JavaScript zwracają wartości przy użyciu słowa kluczowego return. Instrukcji return może być wiele dla jednej funkcji.

Metoda zwracania	Przykład	Opis
Zwracanie prostej wartości	<code>function dodaj(a, b) { return a + b; }</code>	Funkcja zwraca wynik operacji arytmetycznej.
Zwracanie wartości zmiennej	<code>function obliczKwadrat(liczba) { const wynik = liczba * liczba; return wynik; }</code>	Funkcja oblicza kwadrat liczby, przypisuje go do zmiennej i zwraca tę zmienną.
Zwracanie obiektów	<code>function stworzOsobe(imie, wiek) { return { imie: imie, wiek: wiek }; }</code>	Funkcja tworzy i zwraca obiekt z właściwościami imie i wiek.
Zwracanie funkcji	<code>function zewnetrzna() { return function wewnetrzna() { console.log("Witaj Świecie!"); }; }</code>	Funkcja zwraca inną funkcję jako wartość.
Zwracanie tablic	<code>function zwrocTablice() { return [1, 2, 3, 4, 5]; }</code>	Funkcja tworzy i zwraca tablicę z pięcioma elementami.
Zwracanie wartości z warunkami	<code>function porownaj(a, b) { if (a > b) { return "a jest większe"; } else if (a < b) { return "b jest większe"; } else { return "a i b są równe"; } }</code>	Funkcja zwraca różne łańcuchy znaków w zależności od spełnionego warunku.
Zwracanie undefined	<code>function nicNieRob() { }</code>	Funkcja nie ma zadeklarowanego return, więc domyślnie zwraca undefined.

Obiekty

Obiekty

to kolekcja powiązanych danych i funkcjonalności, składająca się z wielu zmiennych i funkcji, które są nazywane właściwościami i metodami obiektu.

Właściwości mogą być wartościami dowolnego typu, takiego jak liczby, łańcuchy znaków czy inne obiekty.

Metody są funkcjami związanymi z obiektem, które mogą manipulować jego właściwościami lub wykonywać zadania związane z obiektem.

```
const prostokat = {
  szerokosc: 5,
  wysokosc: 10,

  // Metoda do obliczania pola powierzchni prostokąta
  obliczPole: function() {
    return this.szerokosc * this.wysokosc;
  },

  // Metoda do obliczania obwodu prostokąta
  obliczObwod: function() {
    return 2 * (this.szerokosc + this.wysokosc);
  }
};

// Użycie metod obiektu prostokat
console.log(`Pole prostokąta: ${prostokat.obliczPole()} jednostek kwadratowych`);
console.log(`Obwód prostokąta: ${prostokat.obliczObwod()} jednostek`);
```

Obiekty

Zalety używania obiektów

Strukturyzacja danych

Obiekty pozwalają na zorganizowanie danych dotyczących jednej "rzeczy" lub "entytetu" w jedną logiczną jednostkę.

Modularyzacja i ponowne wykorzystanie kodu

Metody (funkcje w obrębie obiektów) mogą być używane wielokrotnie dla różnych instancji obiektów, co sprzyja ponownemu użyciu kodu i łatwości jego utrzymania.

Zakapsułkowanie

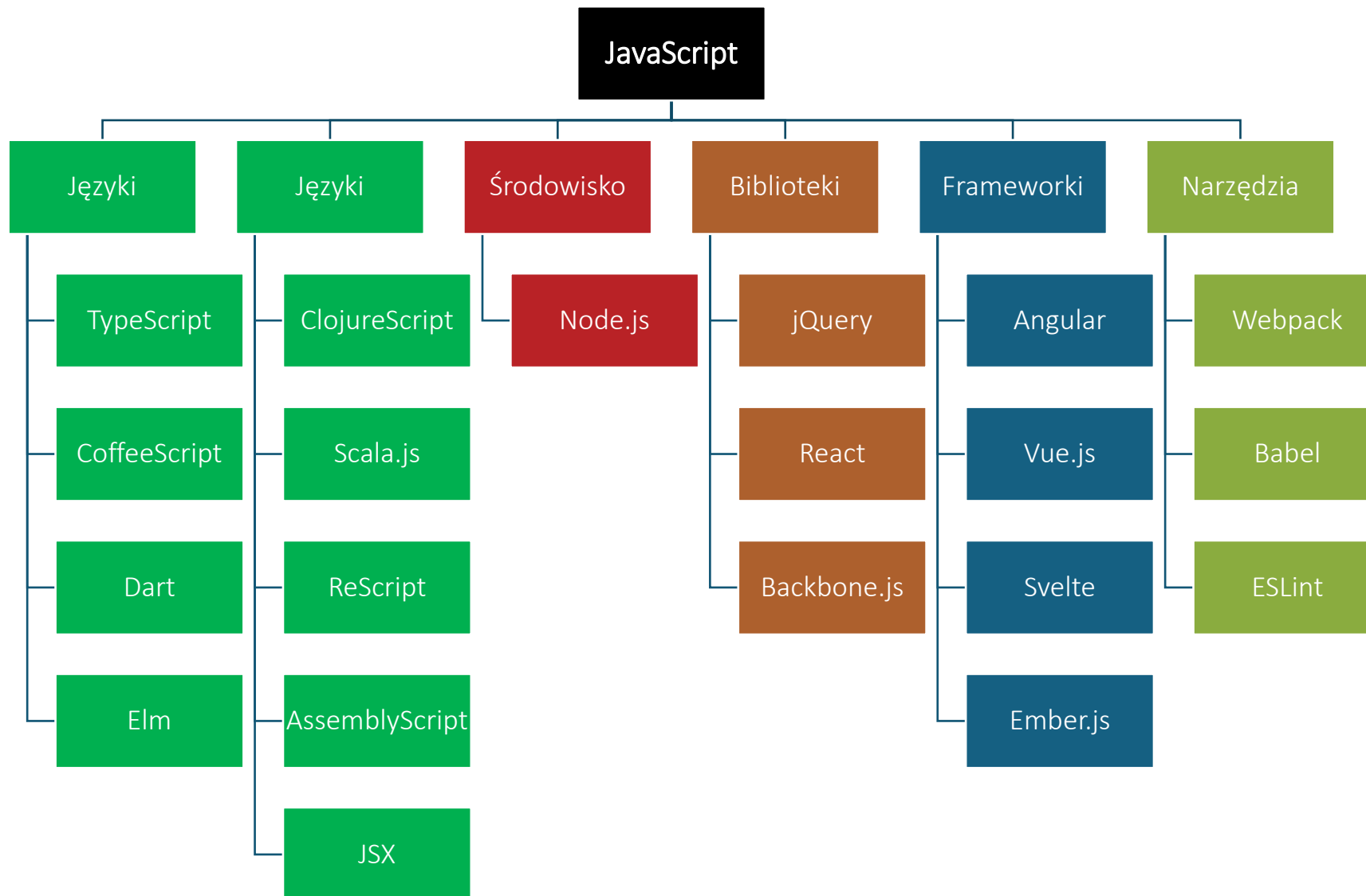
Obiekty umożliwiają zakapsułkowanie związanych ze sobą danych i funkcji, co zmniejsza ryzyko konfliktów w nazwach i ułatwia zarządzanie stanem.

Przedstawianie rzeczywistości

Obiekty są używane do modelowania bytów z rzeczywistego świata, takich jak użytkownicy, zamówienia, produkty itp.

Umożliwiają one przechowywanie informacji w jednym miejscu, ułatwiając manipulowanie danymi i implementowanie logiki biznesowej

Rodzina JavaScript



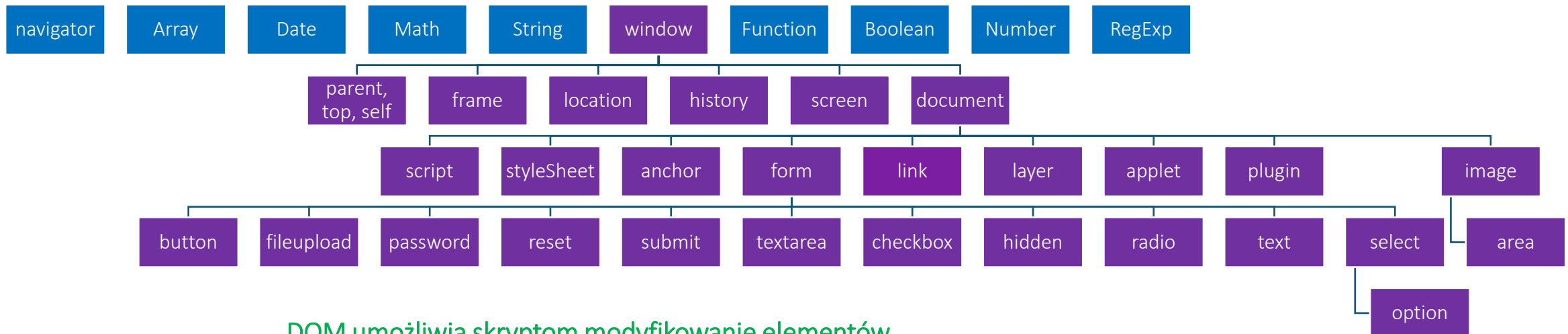
DOM

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)

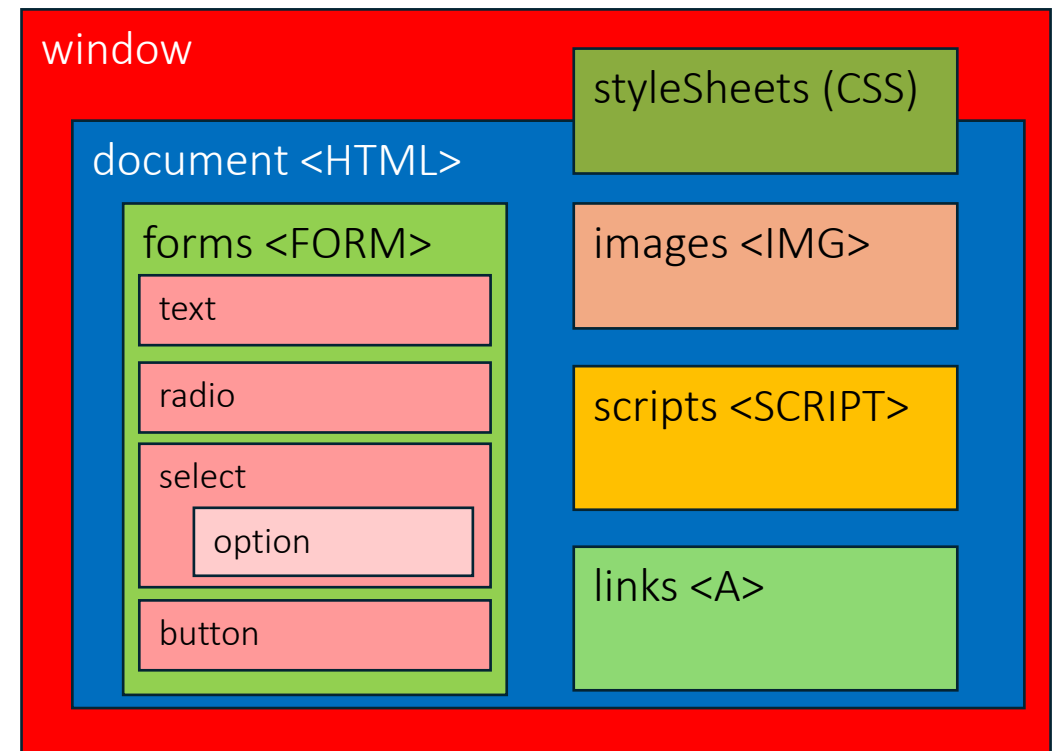
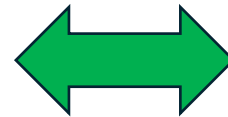
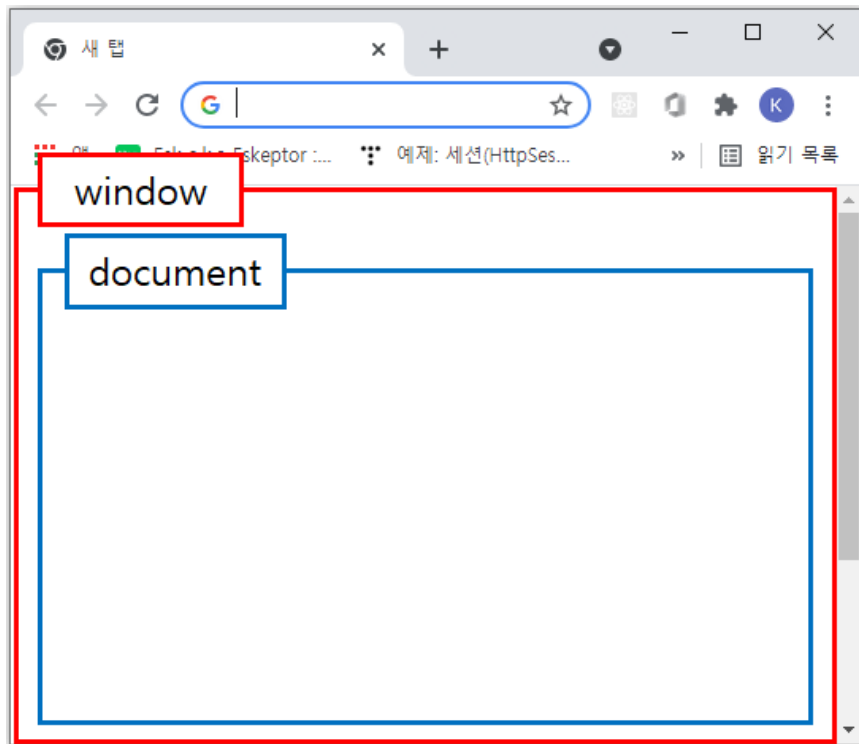
jest to interfejs programowania aplikacji (API) definiujący sposób, w jaki JavaScript może manipulować strukturą i zawartością dokumentu HTML.



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

DOM

DOM (Document Object Model)



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)

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);

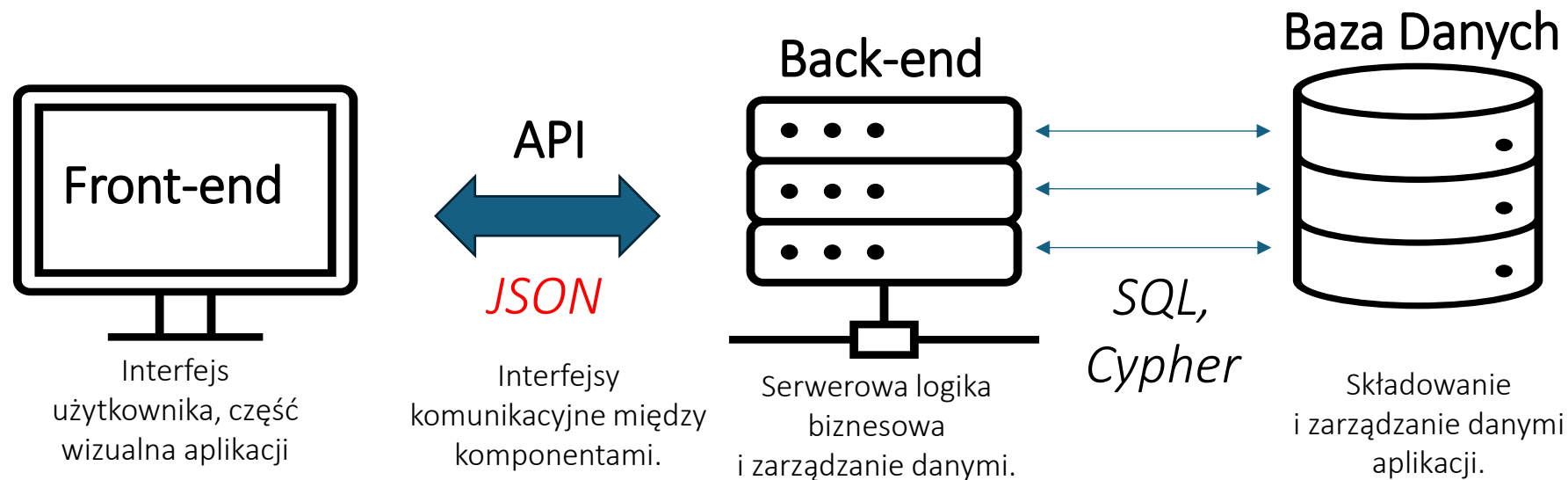
// Ustawianie nowej zawartości HTML wewnątrz <body>
document.body.innerHTML = '<h1>Witaj, świecie!</h1><p>To jest nowy akapit.</p>';
```

JSON

JSON (JavaScript Object Notation)

to lekki format wymiany danych, łatwy do odczytania i zapisywania dla ludzi oraz do parsowania i generowania dla maszyn.

Stanowi aktualnie główny format wymiany danych pomiędzy komponentami systemów



JSON

Budowa

```
{  
  "imie": "Jan",  
  "nazwisko": "Kowalski",  
  "wiek": 30,  
  "miasto": "Warszawa",  
  "znajomi": ["Anna", "Paweł", "Ewa"],  
  "adres": {  
    "ulica": "Krakowska 123",  
    "kod_pocztowy": "00-000",  
    "kraj": "Polska"  
  }  
}
```

- Plik tekstowy z rozszerzeniem .json
- Dane są przedstawione w formie par "klucz"-wartość".
- Obiekty (zestawy danych) są otoczone klamrami {}.
- Klucze są zawsze w podwójnych cudzysłowach (np. "imie").
- Wartości mogą być typu:
 - String
 - Number
 - Object (obiekt)
 - Array (tablica)
 - Boolean
 - Null.
- Tablice są listami wartości otoczone kwadratowymi nawiasami [].

JSON

Praca z JSON w JavaScript

Parsowanie JSON: `JSON.parse()`

służy do konwersji danych w formacie JSON na obiekt JavaScript.

```
const jsonString = '{"imie":"Anna","nazwisko":"Nowak","wiek":28}';
const osoba = JSON.parse(jsonString);

console.log(osoba.imie); // Wyświetli "Anna"
console.log(osoba.wiek); // Wyświetli 28
```

Zamiana obiektu na JSON: `JSON.stringify()`

konwertuje obiekt JavaScript lub wartość na łańcuch tekstowy w formacie JSON

```
const osoba = {
  imie: "Anna",
  nazwisko: "Nowak",
  wiek: 28
};

const jsonString = JSON.stringify(osoba);

console.log(jsonString); // Wyświetli '{"imie":"Anna",'
```

