



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/)

Kolekcje obiektu document

Wykład 04

JĘZYKI PROGRAMOWANIA
STUDIUM PRZYPADKU



Zagadnienia (2h)

- [Kolekcje](#)
- [images](#)
- [links](#)
- [styleSheets](#)
- [scripts](#)
- [forms](#)
- [forms.elements](#)
- [Uniwersalne właściwości i metody obiektów DOM](#)

Zadanie: Raport częściowy z realizacji projektu aplikacji

Raport częściowy z realizacji projektu aplikacji

Opis stanu realizacji projektu ze szczególnym uwzględnieniem implementacji kodu programu i opracowania dokumentacji funkcjonalności systemu (co najmniej 8).

Przedstawienie wyniku

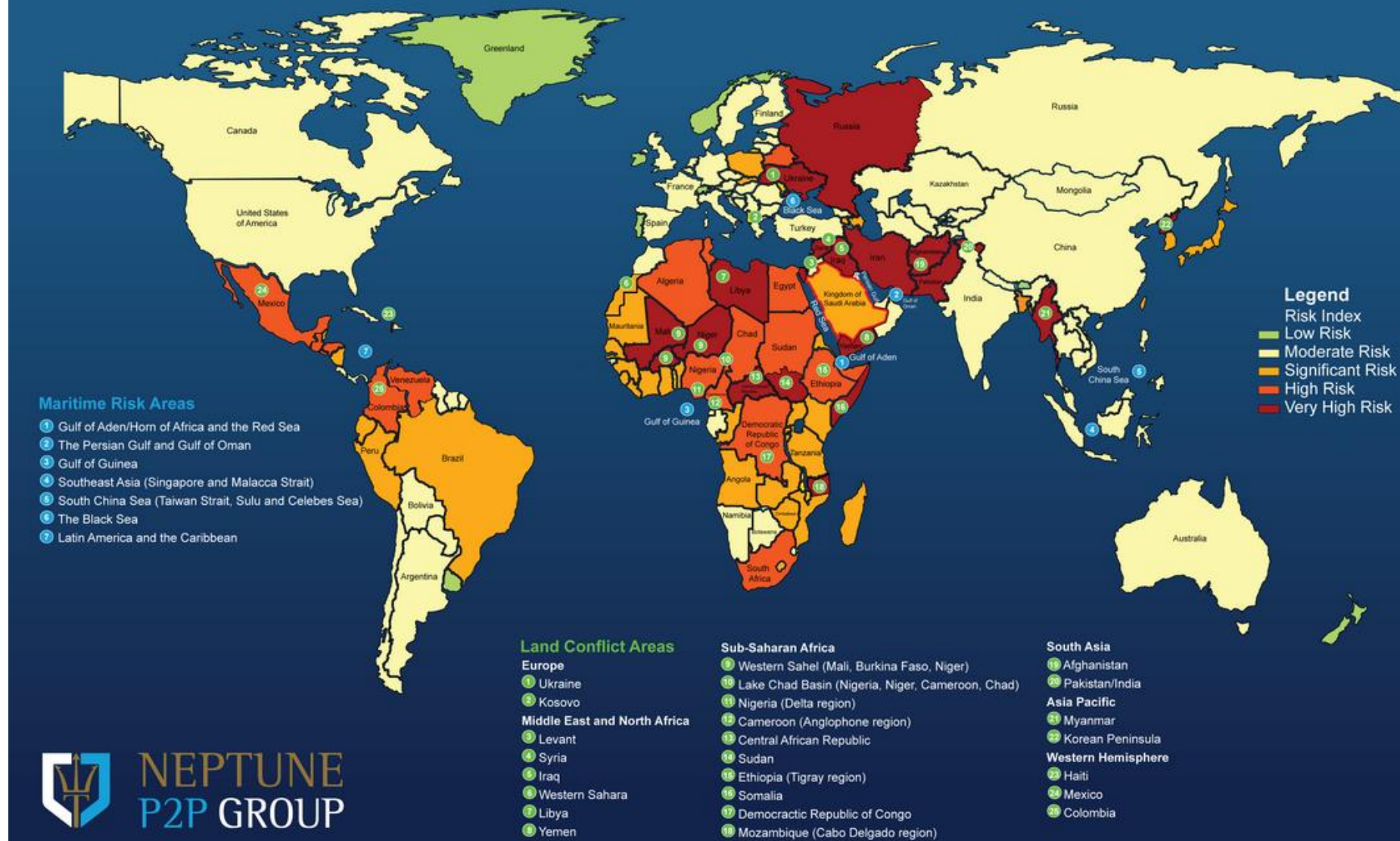
- Dokument tekstowy PDF
- Umieszczenie wyniku w Moodle

Termin

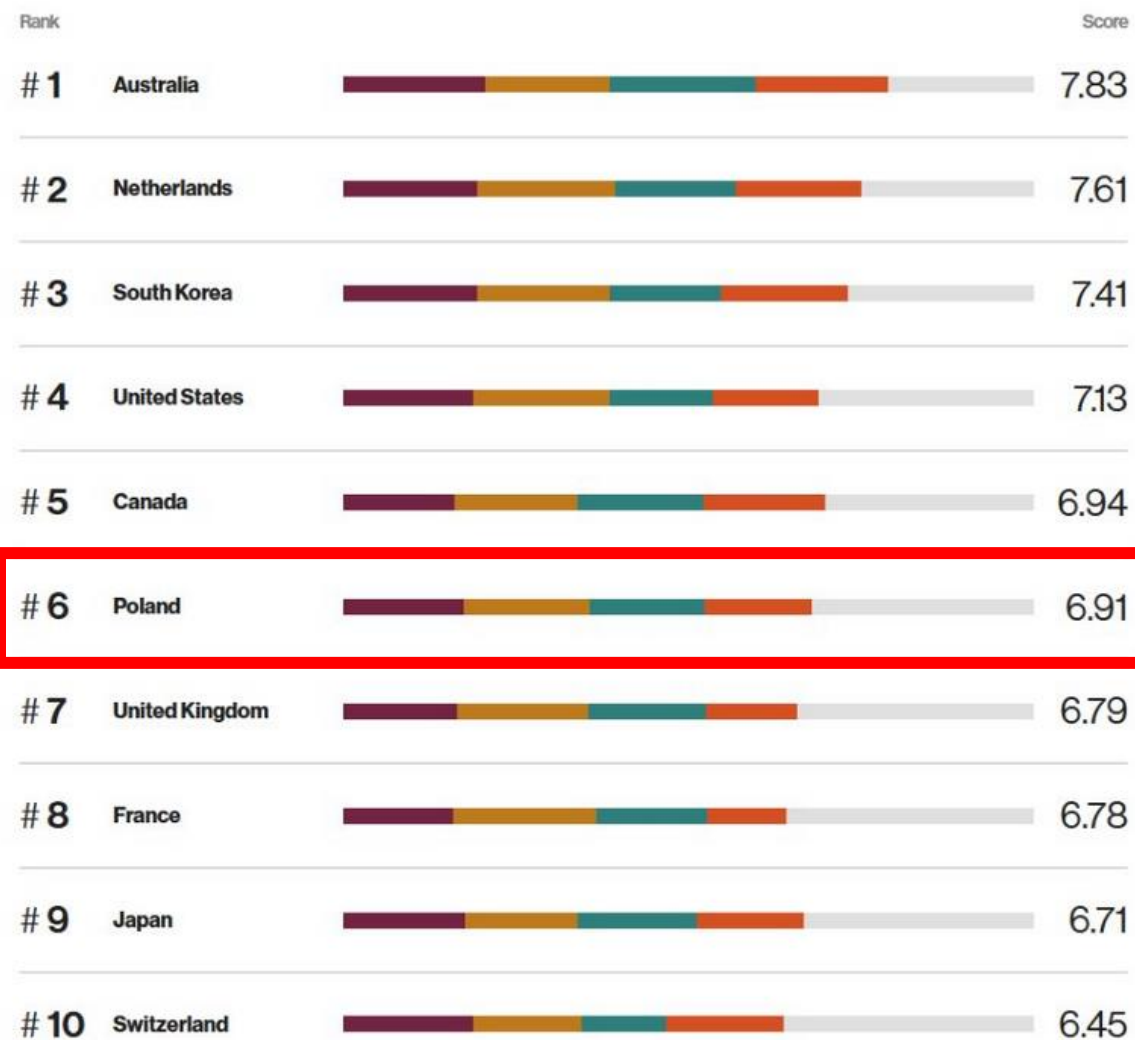
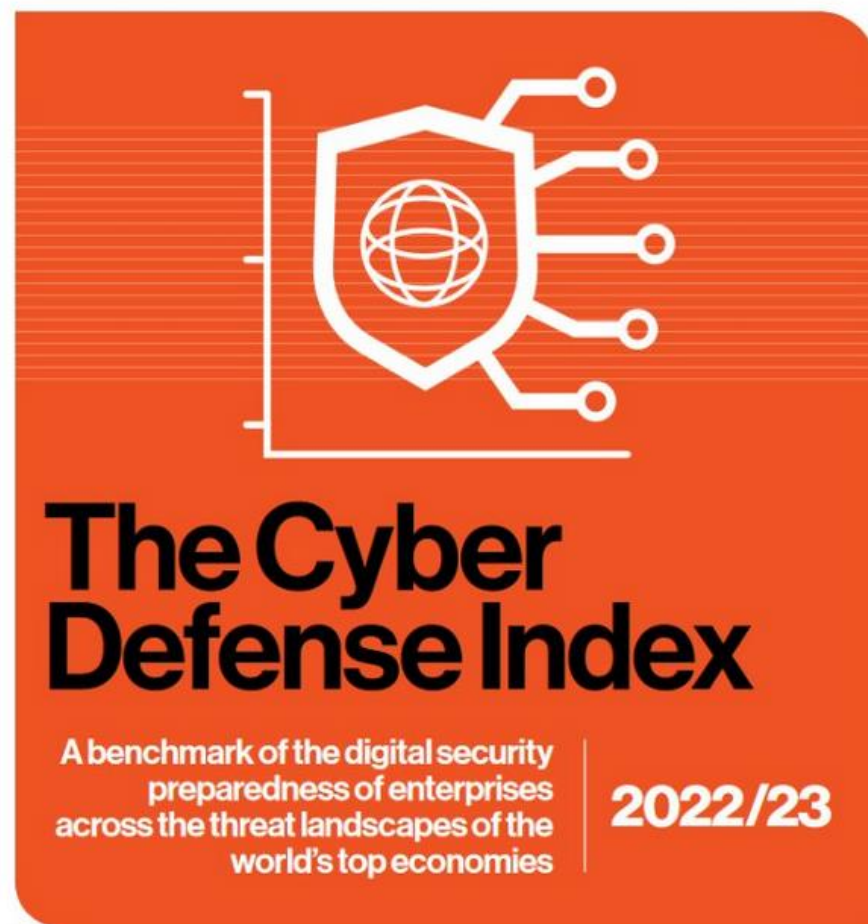
- Podany przy zadaniu w Moodle (około 2 tygodnie)

Global Risk and Security Assessment

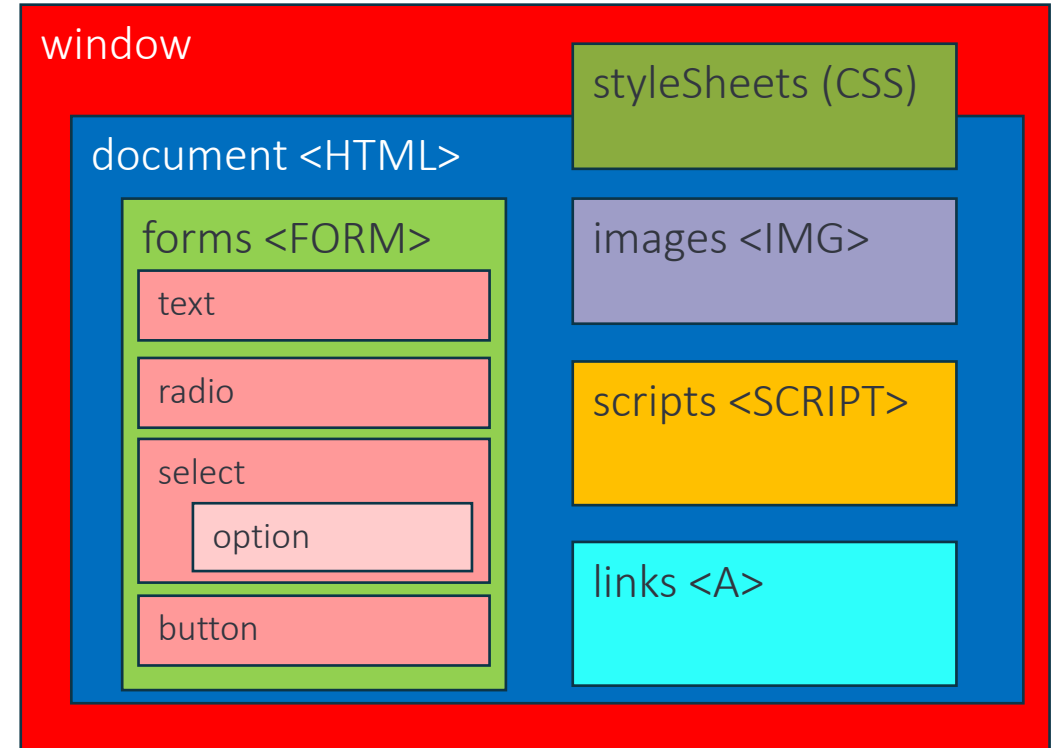
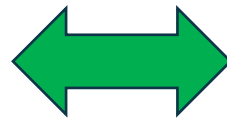
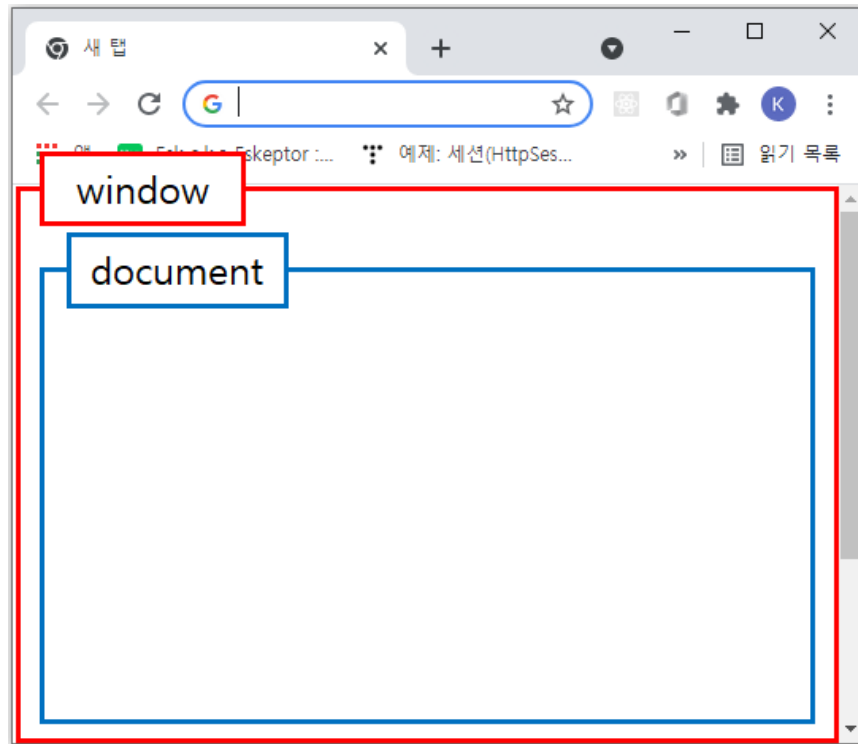
Visualising Maritime and Terrestrial Risk Q4 2023



MIT Technology Review



DOM (Document Object Model)



Kolekcje obiektu document

Kolekcje obiektu document

Kolekcja	Opis
images	Zbiór wszystkich elementów w dokumencie. Umożliwia manipulację i zarządzanie obrazami.
links	Zbiór wszystkich elementów <a> z atrybutem href w dokumencie. Umożliwia manipulację i zarządzanie hiperłączami.
styleSheets	Zbiór wszystkich arkuszy stylów CSS powiązanych z dokumentem, zarówno wewnętrznych, jak i zewnętrznych. Umożliwia manipulację stylami CSS.
scripts	Zbiór wszystkich elementów <script> w dokumencie. Pozwala na dostęp i zarządzanie skryptami.
forms	Zbiór wszystkich formularzy <form> w dokumencie. Umożliwia manipulację i dostęp do danych formularzy.

document.images

document.images

to kolekcja, która zawiera wszystkie elementy znajdujące się w dokumencie.

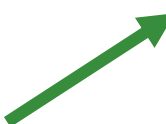
length: Zwraca liczbę obrazków w dokumencie.

item(): Zwraca obrazek określonym indeksem.

namedItem(): Zwraca obrazek z określoną nazwą.

```
// Odczytanie liczby obrazów w dokumencie
console.log(document.images.length);
```

Równorzędne metody dostępu do pojedynczego linku



```
<!-- Obrazy w dokumencie -->



<script>
// Dostęp do obrazów przez indeks
var obrazIndeks = document.images[0];
console.log('Pierwszy obraz w dokumencie:', obrazIndeks.src);

// Dostęp do obrazu po indeksie za pomocą metody item()
var obrazItem = document.images.item(0);
console.log('Obraz uzyskany przez item():', obrazItem.src);

// Dostęp do obrazu po nazwie za pomocą metody namedItem()
var obrazNamedItem = document.images.namedItem('obraz1');
console.log('Obraz uzyskany przez namedItem():', obrazNamedItem.src);
</script>
```

document.areas

document.areas

to kolekcja, która zawiera wszystkie obszary `<area>` związane z obrazkami `<img>` znajdujące się w dokumencie.

href: Adres URL, do którego prowadzi obszar po kliknięciu.

coords: Określa współrzędne obszaru na mapie obrazu.

shape: Określa kształt obszaru na mapie, np. `rect` dla prostokąta, `circle` dla koła, `poly` dla wielokąta.

target: Określa, gdzie otworzyć połączony dokument po kliknięciu na obszar.

```

<map name="mapaPokoju">
  <area shape="rect" coords="34,44,270,350" href="info-o-sofie.html" alt="Sofa">
</map>
```

```
// Dostęp do pierwszego obszaru w dokumencie
var areaElement = document.areas[0];

// Wyświetlenie adresu URL przypisanego do obszaru
console.log(areaElement.href);

// Zmiana kształtu obszaru
areaElement.shape = 'circle';
```

document.images

document.images[0]

src: URL, z którego pochodzi obraz.

alt: Alternatywny tekst obrazu, wyświetlany gdy obraz nie może być załadowany.

width i height: Szerokość i wysokość obrazu w pikselach.

id: Identyfikator elementu obrazu.

className: Lista klas CSS elementu obrazu.

style: Dostęp do stylów inline elementu obrazu.

```
var obraz = document.images[0];

// Właściwości obrazu
console.log('Src:', obraz.src); // URL obrazu
console.log('Alt:', obraz.alt); // Alternatywny tekst obrazu
console.log('Width:', obraz.width); // Szerokość obrazu
console.log('Height:', obraz.height); // Wysokość obrazu
console.log('ID:', obraz.id); // Identyfikator obrazu
console.log('ClassName:', obraz.className); // Lista klas CSS obrazu
console.log('Style:', obraz.style.cssText); // Styl inline obrazu
```

document.images

document.images[0]

onload: Wywoływane, gdy obraz został całkowicie załadowany.

onerror: Wywoływane, gdy obraz nie może zostać załadowany.

onclick: Wywoływane, gdy użytkownik klika na obraz.

onmouseover: Wywoływane, gdy kursor myszy znajduje się nad obrazem.

onmouseout: Wywoływane, gdy kursor myszy opuszcza obraz.

onmousedown: Wywoływane, gdy użytkownik naciska przycisk myszy nad obrazem.

onmouseup: Wywoływane, gdy użytkownik zwalnia przycisk myszy nad obrazem.

onmousemove: Wywoływane, gdy kursor myszy porusza się nad obrazem.

ondragstart: Wywoływane, gdy rozpoczyna się przeciąganie obrazu (jeśli przeciąganie jest dozwolone).

```
var obraz = document.images[0];
obraz.onload = function() { console.log('Obraz został załadowany.');
```

Zmiana obrazka po najechaniu myszą

```

```

document.images

Metody wykorzystania zdarzeń

Metoda	Przykład	Opis	Zalecenia
Atrybut HTML	<pre></pre>	Zdarzenia definiowane bezpośrednio w tagu HTML, gdzie this.src odnosi się do obrazu, na którym wykonano zdarzenie.	Definiowanie zdarzeń bezpośrednio w tagu HTML jest mniej pożądaną praktyką, ponieważ miesza logikę JavaScript z oznaczeniem HTML i może utrudniać utrzymanie kodu.
Funkcja zdarzenia JS	<pre>obraz.onmouseover = function() { console.log('Kursor myszy jest nad obrazem.');</pre>	Zdarzenia przypisane do właściwości obiektu w JavaScript, pozwalające na wykonanie bardziej złożonych operacji.	Metoda zalecana, ponieważ pozwala oddzielić logikę obsługi zdarzeń od struktury HTML

document.links

document.links

to kolekcja, która zawiera wszystkie elementy `<a>` z atrybutem `href` w dokumencie.

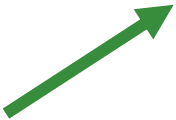
length: Zwraca liczbę linków w dokumencie.

item(): Zwraca link określonym indeksem.

namedItem(): Zwraca link z określoną nazwą.

```
// Odczytanie liczby linków w dokumencie
console.log(document.links.length);
```

Równorzędne metody dostępu do pojedynczego linku



```
<!-- Linki w dokumencie -->
<a href="http://example.com" id="link1" name="link1">Przykładowy Link 1</a>
<a href="http://example.org" id="link2" name="link2">Przykładowy Link 2</a>

<script>
// Dostęp do linków przez nazwę jako indeks
var linkNazwa = document.links['link1'];
console.log('Link uzyskany przez nazwę:', linkNazwa.href);

// Dostęp do linków przez indeks tablicy
var linkIndeks = document.links[0];
console.log('Link uzyskany przez indeks:', linkIndeks.href);

// Dostęp do linku po indeksie za pomocą metody item()
var linkItem = document.links.item(0);
console.log('Link uzyskany przez item():', linkItem.href);

// Dostęp do linku po nazwie za pomocą metody namedItem()
var linkNamedItem = document.links.namedItem('link1');
console.log('Link uzyskany przez namedItem():', linkNamedItem.href);

</script>
```

document.links

document.links[0]

href: URL, do którego prowadzi link.

textContent lub innerText: widoczny tekst linku.

target: określa, gdzie ma zostać otwarty dokument wskazywany przez link.

rel: określa relację pomiędzy bieżącym a wskazanym dokumentem.

id: identyfikator elementu.

className: lista klas CSS elementu.

style: dostęp do stylów inline elementu.

type: typ zasobu, do którego link prowadzi (np. "text/html").

title: tekst podpowiedzi, który wyświetla się po najechaniu kursorem na link..

```
var link = document.links[0];

// Właściwości linku
console.log('Href:', link.href); // URL, do którego prowadzi link
console.log('TextContent:', link.textContent); // Widoczny tekst linku
console.log('Target:', link.target); // Określa, gdzie ma zostać otwarty dokument
console.log('Rel:', link.rel); // Określa relację pomiędzy bieżącym a wskazanym dokumentem
console.log('ID:', link.id); // Identyfikator elementu
console.log('ClassName:', link.className); // Lista klas CSS elementu
console.log('Style:', link.style); // Dostęp do stylów inline elementu
console.log('Type:', link.type); // Typ zasobu, do którego link prowadzi
console.log('Title:', link.title); // Tekst podpowiedzi, który wyświetla się po najechaniu kursorem na link
```

document.links

document.links[0]

onclick: Wywoływane, gdy użytkownik klika na link.

onmouseover: Wywoływane, gdy kursor myszy znajduje się nad linkiem.

onmouseout: Wywoływane, gdy kursor myszy opuszcza link.

onfocus: Wywoływane, gdy link otrzymuje fokus np. przez tabulację lub kliknięcie.

onblur: Wywoływane, gdy link traci fokus.

```
var pierwszyLink = document.links[0];

// Zdarzenie kliknięcia
pierwszyLink.onclick = function() {
    console.log('Kliknięto na link.');
```

```
};

// Zdarzenie najechania kursorem myszy na link
pierwszyLink.onmouseover = function() {
    console.log('Kursor myszy jest nad linkiem.');
```

```
};

// Zdarzenie opuszczenia linku kursorem myszy
pierwszyLink.onmouseout = function() {
    console.log('Kursor myszy opuścił link.');
```

```
};

// Zdarzenie otrzymania fokusu przez link
pierwszyLink.onfocus = function() {
    console.log('Link otrzymał fokus.');
```

```
};

// Zdarzenie utraty fokusu przez link
pierwszyLink.onblur = function() {
    console.log('Link stracił fokus.');
```

```
};
```

document.styleSheets

document.styleSheets

to kolekcja, która zawiera wszystkie elementy arkuszy styli <style> lub <link> znajdujące się w dokumencie.

length: Zwraca liczbę arkuszy stylów w dokumencie.

item(): Zwraca arkusz stylów określonym indeksem.

namedItem(): Zwraca arkusz stylów z określoną nazwą.

```
// Odczytanie liczby arkuszy stylów w dokumencie
console.log(document.styleSheets.length);
```

Równorzędne metody dostępu do pojedynczego arkusza styli

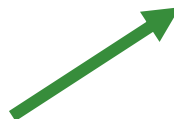
```
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Przykład różnych deklaracji stylów</title>
  <style id="wewnetrzneStyle">
    body { background-color: #f0f0f0; }
    p { color: red; }
  </style>
  <link rel="stylesheet" id="arkusz2" href="styles2.css">
</head>
<body>

<script>
  // Dostęp do arkusza stylów przez indeks
  var arkuszIndeks = document.styleSheets[0];
  console.log('Pierwszy arkusz stylów w dokumencie:', arkuszIndeks.href);

  // Dostęp do arkusza stylów po indeksie za pomocą metody item()
  var arkuszItem = document.styleSheets.item(0);
  console.log('Arkusz stylów uzyskany przez item():', arkuszItem.href);

  // Dostęp do arkusza stylów po id za pomocą metody getElementById()
  var arkuszNamedItem = document.getElementById('wewnetrzneStyle').sheet;
  console.log('Arkusz stylów wewnętrzny:', arkuszNamedItem ? arkuszNamedItem.href : '');

  // Dostęp do zewnętrznego arkusza stylów przez id
  var zewnetrznyArkusz = document.getElementById('arkusz2').sheet;
  console.log('Zewnętrzny arkusz stylów:', zewnetrznyArkusz.href);
</script>
</body>
</html>
```



document.styleSheets

document.styleSheets[0]

disabled: Właściwość disabled umożliwia wyłączenie lub włączenie arkusza stylów.

href: Atrybut href zwraca URL arkusza stylów, jeśli został załączony za pomocą elementu <link>.

media: Właściwość media zwraca typy mediów, dla których arkusz stylów ma być stosowany.

title: Tytuł arkusza stylów, który może być używany przez przeglądarki do zmiany arkuszy stylów.

```
var arkuszIndeks = document.styleSheets[0];

// Log properties of the first stylesheet
console.log('Href:', arkuszIndeks.href); // URL of the linked stylesheet
console.log('Type:', arkuszIndeks.type); // Type/MIME of the stylesheet
console.log('Title:', arkuszIndeks.title); // Title of the stylesheet
console.log('Media:', arkuszIndeks.media.mediaText); // Media type of the stylesheet
console.log('Disabled:', arkuszIndeks.disabled); // Boolean indicating if the styles
```

Wartość <u>media</u>	Opis	Przykład użycia w CSS
"screen"	Domyślna wartość, stosowana gdy nie ma zdefiniowanych zapytań multimedialnych w arkuszu stylów. Styl jest stosowany do wyświetlania na ekranach.	@media screen { /* Style dla ekranów */ }
"all"	Stosowana gdy w zapytaniu multimedialnym występuje typ multimedialny all. Styl jest stosowany we wszystkich typach mediów.	@media all { /* Style dla wszystkich typów mediów */ }
"(max-width: 768px)"	Stosowana gdy ekran ma szerokość mniejszą lub równą 768 pikseli.	@media (max-width: 768px) { /* Style dla ekranów o szerokości mniejszej lub równej 768px */ }
"not all"	Wskazuje, że w arkuszu stylów zdefiniowano zapytania multimedialne.	Brak

document.scripts

document.scripts

To kolekcja, która zawiera wszystkie elementy <script> znajdujące się w dokumencie..

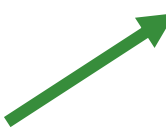
length: Zwraca liczbę skryptów w dokumencie.

item(): Zwraca skrypt o określonym indeksie.

namedItem(): Zwraca skrypt z określoną nazwą.

```
// Odczytanie liczby skryptów w dokumencie
console.log(document.scripts.length);
```

Równorzędne metody dostępu do pojedynczego skryptu



```
<script id="skrypt1" src="skrypt.js"></script>
<script id="skrypt2" type="module" src="modul.js"></script>

<script>
  // Odczytanie liczby skryptów w dokumencie
  console.log(document.scripts.length);

  // Dostęp do skryptu przez indeks tablicy
  var skryptIndeks = document.scripts[0];
  console.log('Skrypt uzyskany przez indeks:', skryptIndeks.src);

  // Dostęp do skryptu po indeksie za pomocą metody item()
  var skryptItem = document.scripts.item(0);
  console.log('Skrypt uzyskany przez item():', skryptItem.src);

  // Dostęp do skryptu po nazwie za pomocą metody namedItem()
  var skryptNamedItem = document.scripts.namedItem('skrypt1');
  console.log('Skrypt uzyskany przez namedItem():', skryptNamedItem.src);
</script>
```

document.scripts

document.scripts[0]

src: URL skryptu (jeśli został załadowany z zewnętrznego źródła).

text: Zawartość tekstowa wewnętrzna elementu <script>, jeśli skrypt jest wbudowany.

type: Typ MIME skryptu. Domyślnie "text/javascript".

async: Boolowska wartość wskazująca, czy skrypt powinien być wykonywany asynchronicznie. Domyślnie „false”.

defer: Boolowska wartość wskazująca, czy wykonanie skryptu powinno być odroczone. Domyślnie „false”.

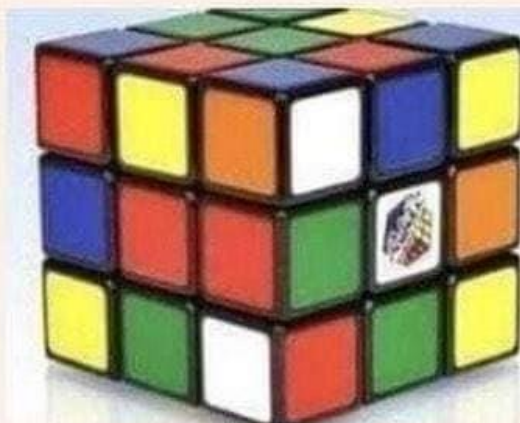
```
let pierwszySkrypt = document.scripts[0];

// Wypisanie URL skryptu
console.log(pierwszySkrypt.src);

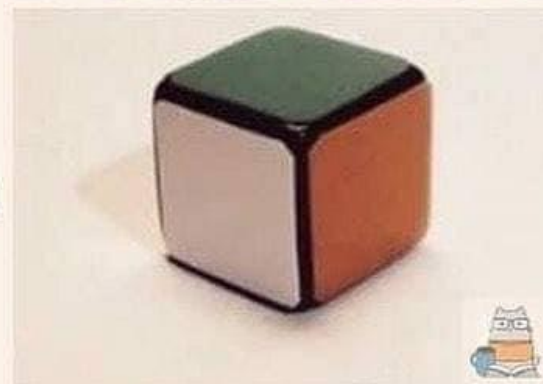
// Sprawdzenie, czy skrypt jest asynchroniczny
console.log(pierwszySkrypt.async);

// Zmiana zawartości skryptu
pierwszySkrypt.text = 'console.log("Nowy skrypt");';
```

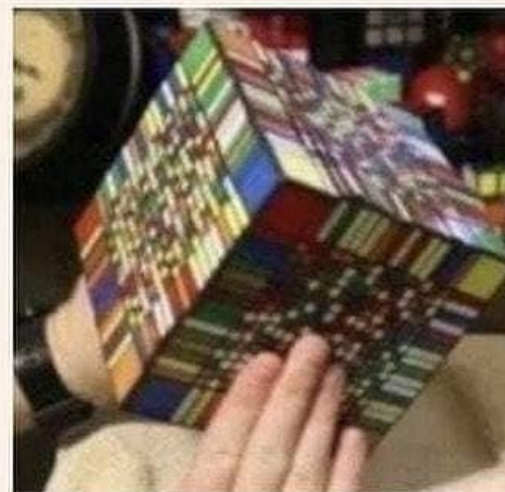
Co omówił
wykładowca



Co
zapamiętałem



Co było
na zaliczeniu



document.forms

document.forms

to kolekcja, która zawiera wszystkie elementy <form> znajdujące się w dokumencie.

length: Zwraca liczbę formularzy w dokumencie.

elements: Dostarcza dostęp do wszystkich elementów wybranego formularza.

item(): Zwraca formularz o określonym indeksie.

namedItem(): Zwraca formularz z określoną nazwą.

```
// Odczytanie liczby formularzy w dokumencie
console.log(document.forms.length);
```

Równorzędne metody dostępu do pojedynczego formularza



```
<form id="formularz1" name="formularz1">
  <label for="imie">Imię:</label>
  <input type="text" id="imie" name="imie">
  <input type="submit" value="Wyślij">
</form>

<script>
// Dostęp do formularza przez nazwę jako indeks
var formularzNazwa = document.forms['formularz1'];
console.log('Formularz uzyskany przez nazwę:', formularzNazwa);

// Dostęp do formularza przez indeks tablicy
var formularzIndeks = document.forms[0];
console.log('Formularz uzyskany przez indeks:', formularzIndeks);

// Dostęp do formularza po indeksie za pomocą metody item()
var formularzItem = document.forms.item(0);
console.log('Formularz uzyskany przez item():', formularzItem);

// Dostęp do formularza po nazwie za pomocą metody namedItem()
var formularzNamedItem = document.forms.namedItem('formularz1');
console.log('Formularz uzyskany przez namedItem():', formularzNamedItem);

</script>
```

document.forms

document.forms['formularz1']

elements: Zwraca kolekcję elementów formularza.

length: Zwraca liczbę elementów w formularzu.

action: Adres URL, do którego formularz jest przesyłany.

method: Metoda HTTP używana do przesłania formularza, np. "GET" lub "POST".

name: Nazwa formularza.

target: Określa gdzie wyświetlić odpowiedź po przesłaniu formularza.

```
// Dostęp do formularza
var formularz = document.forms['formularz1'];

// elements: Zwraca kolekcję elementów formularza
var elementy = formularz.elements;
console.log('Elementy formularza:', elementy);

// length: Zwraca liczbę elementów w formularzu
var liczbaElementow = formularz.elements.length;
console.log('Liczba elementów w formularzu:', liczbaElementow);

// action: Adres URL, do którego formularz jest przesyłany
var akcja = formularz.action;
console.log('Akcja formularza (URL):', akcja);

// method: Metoda HTTP używana do przesłania formularza
var metoda = formularz.method;
console.log('Metoda formularza:', metoda);

// name: Nazwa formularza
var nazwa = formularz.name;
console.log('Nazwa formularza:', nazwa);

// target: Określa gdzie wyświetlić odpowiedź po przesłaniu formularza
var cel = formularz.target;
console.log('Cel formularza (target):', cel);
```

document.forms

document.forms['formularz1']

onsubmit: Wywoływane, gdy formularz jest przesyłany.

onreset: Wywoływane, gdy formularz jest resetowany.

oninput: Wywoływane przy każdej zmianie wartości inputów formularza.

onchange: Wywoływane, gdy zmienia się wartość jednego z elementów formularza (np. po stracie focusa).

```
var formularz = document.forms['formularz1'];

formularz.onsubmit = function(event) {
    event.preventDefault();
    var imie = formularz.elements['imie'].value;
    var email = formularz.elements['email'].value;
    console.log('Imię: ', imie);
    console.log('Email: ', email);
    // Tutaj możesz dodać kod do obsługi wysyłania formularza
};

formularz.onreset = function() {
    console.log('Formularz został zresetowany.');
```

// Tutaj możesz dodać kod do obsługi resetowania formularza

```
};

// Dostęp do pola imię poprzez właściwość elements
var inputImie = formularz.elements['imie'];

inputImie.oninput = function() {
    console.log('Aktualna wartość imienia: ', this.value);
    // Tutaj możesz dodać kod do obsługi zmiany wartości w polu imię
};

formularz.elements['email'].onchange = function() {
    console.log('Nowy adres email: ', this.value);
    // Tutaj możesz dodać kod do obsługi zmiany wartości w polu email
};
```



document.forms

document.forms['formularz1']

submit(): Przesyła formularz.

reset(): Resetuje wszystkie elementy formularza do ich domyślnych wartości.

checkValidity(): Sprawdza, czy wszystkie pola formularza spełniają założenia walidacji.

```
// Dostęp do formularza
var formularz = document.forms['formularz1'];

// Funkcja do przesyłania formularza
function przeslijFormularz() {
    if (formularz.checkValidity()) {
        formularz.submit();
    } else {
        console.log('Formularz zawiera błędy.');
    }
}

// Funkcja do resetowania formularza
function resetujFormularz() {
    formularz.reset();
}
```

document.forms

formularz.checkValidity()

walidacja odbywa się na podstawie atrybutów walidacyjnych, które można dodać do elementów HTML formularza.

Metoda ta zwraca true, jeżeli element spełnia wszystkie założenia walidacji, lub false, jeśli walidacja nie jest spełniona.

Oto najważniejsze atrybuty:

required: Określa, że pole formularza musi zostać wypełnione przed wysłaniem.

type: Definiuje typ danych, które mają być wprowadzone, np. email, number, date, itp.

min i max: Dla typów liczbowych, określa minimalną i maksymalną dopuszczalną wartość.

minlength i maxlength: Dla typów tekstowych, określa minimalną i maksymalną liczbę znaków.

pattern: Definiuje wyrażenie regularne, z którym wprowadzone dane muszą się zgadzać.

```
<form id="formularz1">
  <input type="text" id="imie" name="imie" required minlength="2">
  <input type="email" id="email" name="email" required>
  <input type="submit" value="Wyślij">
</form>

<script>
  var formularz = document.getElementById('formularz1');
  formularz.onsubmit = function(event) {
    if (!formularz.checkValidity()) {
      event.preventDefault(); // Zapobiega wysłaniu formularza
      console.log('Formularz zawiera błędy.');
    }
  };
</script>
```

forms.elements

`document.forms['formularz1'].elements['pole1']`

odnosi się do kolekcji kontrolek formularza, które znajdują się w obiekcie form.

value: Przechowuje lub ustawia wartość elementu.

form: Wskazuje na formularz, do którego element należy.

name: Nazwa elementu.

type: Typ elementu, np. "text", "button" itp.

disabled: Określa, czy element jest wyłączony.

focus(): Ustawia fokus na wybranym elemencie.

blur(): Usuwa fokus z wybranego elementu.

```
<form id="formularz1" name="formularz1">
  <label for="pole1">Pole 1:</label>
  <input type="text" id="pole1" name="pole1" value="Wpisz tekst tutaj">
  <input type="submit" value="Wyślij">
</form>

// Dostęp do konkretnego elementu formularza
var pole = document.forms['formularz1'].elements['pole1'];

// Właściwości
console.log(pole.value);    // Wyświetla wartość elementu
console.log(pole.form);    // Wyświetla formularz, do którego element należy
console.log(pole.name);    // Wyświetla nazwę elementu
console.log(pole.type);    // Wyświetla typ elementu
console.log(pole.disabled); // Sprawdza, czy element jest wyłączony

// Funkcja ustawiająca fokus na pole tekstowe
function ustawFocus() {
  document.forms['formularz1'].elements['pole1'].focus();
}

// Funkcja usuwająca fokus z pola tekstowego
function usunFocus() {
  document.forms['formularz1'].elements['pole1'].blur();
}
```

forms.elements

Typ pola	Opis	Przykład HTML	Przykład JavaScript
text	Pole tekstowe do wprowadzania krótkiego tekstu	<code>input type="text" id="imie" name="imie" value="Jan"</code>	<pre>const imieInput = document.getElementById('imie'); console.log('Wprowadzono tekst:', imieInput.value);</pre>
password	Pole hasła, w którym wprowadzony tekst jest maskowany	<code>input type="password" id="haslo" name="haslo"</code>	<pre>const hasloInput = document.getElementById('haslo'); console.log('Wprowadzono hasło:', hasloInput.value);</pre>
checkbox	Pole wyboru umożliwiające zaznaczenie lub odznaczenie opcji	<code>input type="checkbox" id="zgoda" name="zgoda" value="true"</code>	<pre>const zgodaCheckbox = document.getElementById('zgoda'); console.log('Zaznaczono:', zgodaCheckbox.checked);</pre>
radio	Przycisk opcji umożliwiający wybór jednej opcji z grupy	<code>div class="radio-group"> <input type="radio" id="meczyczna" name="plec" value="meczyczna"> <label for="meczyczna">Mężczyzna</label> <input type="radio" id="kobieta" name="plec" value="kobieta"> <label for="kobieta">Kobieta</label> </div></code>	<pre>const radioButton = document.querySelector('input[name="plec"]'); console.log('Wybrano:', radioButton.value);</pre>
select	Lista rozwijana umożliwiająca wybór jednej opcji z wielu	<code>select id="kraj" name="kraj"> <option value="Polska">Polska</option> <option value="Niemcy">Niemcy</option> </select></code>	<pre>const krajSelect = document.getElementById('kraj'); console.log('Wybrano kraj:', krajSelect.value);</pre>
textarea	Pole tekstowe do wprowadzania dłuższego tekstu	<code>textarea id="komentarz" name="komentarz" rows="5" cols="30"></textarea></code>	<pre>const komentarzTextarea = document.getElementById('komentarz'); console.log('Wprowadzono tekst:', komentarzTextarea.value);</pre>
button	Przycisk do wykonywania różnych akcji	<code>input type="button" id="wyslij" value="Wyślij"</code>	<pre>const wyslijButton = document.getElementById('wyslij'); wyslijButton.addEventListener('click', () => { console.log('Kliknięto przycisk "Wyślij"'); });</pre>
submit	Przycisk do wysłania formularza	<code>input type="submit" id="submit" value="Wyślij"</code>	<pre>// Przykład użycia znajduje się w kodzie dla typu "button"</pre>
reset	Przycisk do zresetowania formularza	<code>input type="reset" id="reset" value="Wyczyść"</code>	<pre>const resetButton = document.getElementById('reset'); resetButton.addEventListener('click', () => { console.log('Kliknięto przycisk "Wyczyść"'); form.reset(); });</pre>
file	Pole umożliwiające przesłanie pliku z komputera użytkownika	<code>input type="file" id="plik" name="plik"</code>	<pre>const plikInput = document.getElementById('plik'); plikInput.addEventListener('change', (event) => { const file = event.target.files[0]; if (file) { console.log('Wybrano plik:', file.name); } });</pre>

forms.elements

Dla elementów typu input i textarea

select(): Wybiera cały tekst w elemencie input lub textarea.

setCustomValidity(message): Umożliwia ustawienie własnej wiadomości walidacji.

Dla elementów typu checkbox i radio

checked: Wartość boolowska określająca, czy pole wyboru lub radio jest zaznaczone.

```
<form id="formularz">
  <input type="text" id="textInput" value="Zaznacz mnie">
  <input type="checkbox" id="checkInput">
  <input type="submit" value="Wyślij">
</form>

<script>
var textInput = document.getElementById("textInput");
textInput.select();
textInput.setCustomValidity("To pole jest wymagane.");

var checkInput = document.getElementById("checkInput");
checkInput.checked = true;

document.getElementById("formularz").onsubmit = function(event) {
  event.preventDefault();
  console.log('Pole tekstowe: ' + textInput.value, 'Checkbox: ' + checkInput.checked);
};
</script>
```

forms.elements

Dla elementu typu option

selected: Wartość boolowska określająca, czy opcja jest wybrana.

text: Tekst widoczny dla użytkownika

Dla elementu typu select

options: Kolekcja zawierająca elementy option, które są dziećmi elementu select.

selectedIndex: Indeks wybranego elementu option.

multiple: Wartość boolowska określająca, czy z elementu select można wybrać więcej niż jedną opcję.

```
<select id="wybor" multiple>
  <option value="opcja1">Opcja 1</option>
  <option value="opcja2" selected>Opcja 2</option>
  <option value="opcja3">Opcja 3</option>
</select>

<script>
  var wybor = document.getElementById("wybor");

  console.log(`Indeks wybranej opcji: ${wybor.selectedIndex}`);
  console.log(`Czy można wybrać wiele opcji: ${wybor.multiple}`);
  Array.from(wybor.options).forEach(opt => {
    if (opt.selected) console.log(`Wybrano opcję: ${opt.text}`);
  });
</script>
```

Uniwersalne właściwości i metody obiektów DOM

Metoda	Opis	Przykład użycia
innerHTML	Pozwala na odczytanie lub zmianę zawartości HTML wewnątrz elementu.	<pre>const element = document.getElementById('mojElement'); console.log(element.innerHTML); element.innerHTML = 'Nowa zawartość HTML';</pre>
style	Umożliwia dostęp i modyfikację stylów CSS elementu bezpośrednio z poziomu JavaScript.	<pre>const element = document.getElementById('mojElement'); console.log(element.style.backgroundColor); element.style.backgroundColor = 'red';</pre>
classList	Ułatwia manipulację klasami CSS elementu, np. dodawanie, usuwanie lub przełączanie klas.	<pre>const element = document.getElementById('mojElement'); console.log(element.classList.contains('mojaKlasa')); element.classList.add('nowaKlasa'); element.classList.remove('mojaKlasa');</pre>
getAttribute() / setAttribute()	Służą do odczytu i ustawiania wartości atrybutów elementu.	<pre>const element = document.getElementById('mojElement'); const identyfikator = element.getAttribute('id'); element.setAttribute('data-role', 'przycisk');</pre>
addEventListener()	Rejestruje funkcję, która zostanie wywołana za każdym razem, gdy określone zdarzenie zostanie wyzwolone na elemencie.	<pre>const element = document.getElementById('mojElement'); element.addEventListener('click', () => { console.log('Kliknięto element!');});</pre>
querySelector() / querySelectorAll()	Pozwala na wyszukiwanie elementów w dokumencie zgodnie z selektorami CSS. querySelector() zwraca pierwszy pasujący element, a querySelectorAll() zwraca wszystkie pasujące elementy jako NodeList.	<pre>const element = document.querySelector('#mojElement'); const elementy = document.querySelectorAll('.mojaKlasa');</pre>
hasAttribute()	Sprawdza, czy element posiada określony atrybut.	<pre>const element = document.getElementById('mojElement'); console.log(element.hasAttribute('data-role'));</pre>

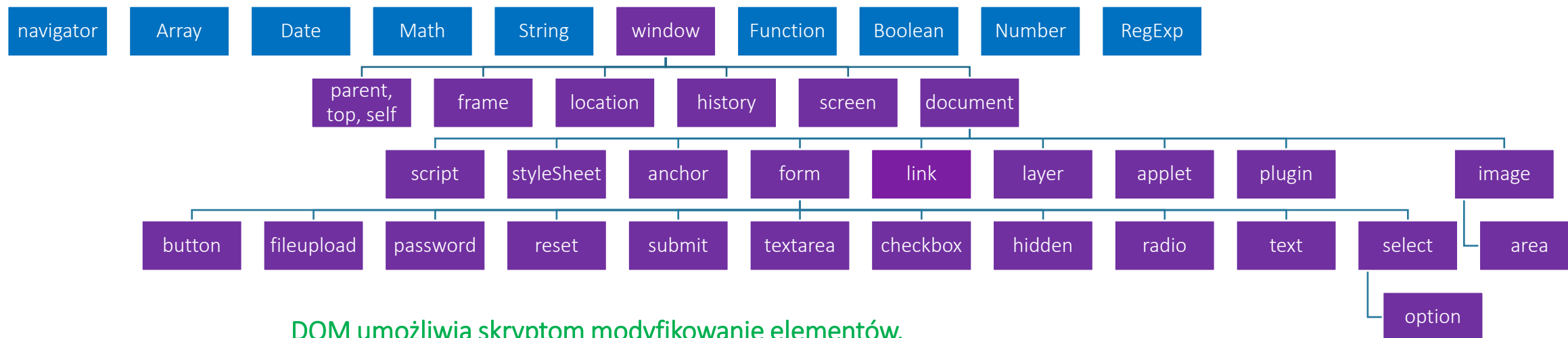
Podsumowanie

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)

Dodatkowe informacje

Zasoby Internetu

- DOM
 - <https://kursjs.pl/kurs/dom/dom>
 - https://www.w3schools.com/js/js_htmlDOM.asp
 - https://youtu.be/LieZnFj7oPU?si=F2cFY_jlEhWlCoFL (film)



How users see the IT guy



How the IT guy see the users

