Tworzenie Stron Internetowych

odcinek 11

JavaScript

JavaScript (ECMAScript) – skryptowy język programowania powszechnie używany w Internecie. Skrypty JS dodają do stron WWW interaktywność i funkcjonalności, np.: reagowanie na zdarzenia generowane przez użytkownika, sprawdzanie poprawności formularzy, dynamiczne modyfikowanie zawartości strony i in.

Początki JS sięgają 1995 roku (Brendan Eich, Netscape). Za standaryzację odpowiada Ecma International. Standard języka jest opisany w specyfikacji ECMA-262, która definiuje język ECMAScript.

Skrypty JS nie wymagają kompilacji. Ich używanie jest darmowe.

Możliwe jest też tworzenie w JS zwykłych aplikacji komputerowych, mobilnych, serwerowych i desktopowych.

JavaScript to nie to samo co Java. To są dwa różne języki programowania.

JS – edytory

Edycja plików JS

Skrypt JS to plik tekstowy, można więc go pisać w dowolnym edytorze tekstowym. Warto jednak używając edytorów programistycznych, które ułatwiają pisanie kodu. Dobry edytor JS posiada:

- kolorowanie składni (unikanie błędów w poleceniach),
- zaawansowana edycja (wielopoziomowe cofanie, znajdź-zastąp,…),
- generatory elementów JS .

Często edytory do tworzenia stron internetowych obsługują edycję dokumentów HTML, CSS i JS.

JS – zalety

Dlaczego używać JS:

- **składnia JS jest prosta** – twórcy stron internetowych mogą wykorzystać funkcjonalność JS nie będąc programistami
- **JS zwiększa interakcję z użytkownikiem** – skrypty mogą reagować na zdarzenia generowane po stronie użytkownika
- **JS czyta i zapisuje elementy HTML** – skrypty mogą odczytać i zmienić zawartość dowolnego elementu HTML oraz je ukrywać, pokazywać, kasować, tworzyć i powielać. Mogą też zmieniać atrybuty elementów HTML oraz ich styl CSS.
- **JS może sprawdzać poprawność danych** – skrypty pozwalają sprawdzić dane wpisane w formularz przed ich wysłaniem, co oszczędza pracy serwerowi
- **JS wykryje z jakiej przeglądarki korzysta użytkownik** – skrypt wykrywając typ przeglądarki może załadować wersję strony zaprojektowaną dla niej
- **JS może tworzyć ciasteczka (cookies)** – skrypt pozwala przetrzymywać i odczytywać informacje na komputerze użytkownika
- **skrypty JS możemy pisać sami lub wykorzystać gotowe**, pobrane z sieci (np.: www.kurshtml.edu.pl/skrypty, www.w3schools.com/js/js_project_counter.asp)

JS – osadzanie

Skrypty JS umieszczane są w dokumencie HTML (<head> lub/i <body>) za pomocą znacznika **<script>** w postaci:

- samego kodu JS

```
<script type="text/javascript">
/* <![CDATA[ */
 kod JS
/*]]> */</script>
```

obecnie atrybut type nie jest wymagany

zabezpiecza przed interpretacją kodu jako składni HTML, tylko dla xHTML

- odnośnika do pliku zawierającego kod JS

```
<script type="text/javascript" src="plik.js"></script>
```

W jednym dokumencie HTML można umieścić dowolną liczbę skryptów.

W części **<head>** dokumentu HTML można podać deklarację typu użytego języka skryptowego, ale nie jest to konieczne, bo JS jest domyślnym językiem skryptowym w HTML:

```
<meta http-equiv="Content-Script-Type" content="text/javascript">
```

Kolejność osadzania skryptów ma znaczenie, ponieważ wykonywane są one w takiej kolejności, w jakiej zostały umieszczone w kodzie strony.

Jeśli przeglądarka użytkownika nie obsługuje JS, to nie będzie miał on dostępu do treści/funkcji generowanych przez skrypt JS. O tej sytuacji możemy użytkownika poinformować używając znacznika **<noscript>**:

```
<noscript>Uwaga, Twoja przeglądarka nie obsługuje JavaScript!</noscript>
```

JS – osadzanie

Zalety umieszczania skryptów JS osobnych plikach:

- ten sam kod można wykorzystać do wielu stron WWW
- oddzielenie kodu HTML i JS
- ułatwienie edycji kodów
- szybsze ładowanie stron WWW

Jest też możliwość użycia zewnętrznych skryptów przez podanie ich adresu URL

```
<script src="https://www.jakas-strona.com/js/skryptJS-1.js"></script>
```

JS – składnia

Ogólnie o składni

- Kod (skrypt) JS składa się z zestawu instrukcji (statements). **Poszczególne instrukcje oddzielane są od siebie średnikiem.**
- Instrukcje JavaScript składają się z: wartości (values), operatorów (operators), wyrażeń (expressions), słów kluczowych (keywords) i komentarzy (comments).
- Instrukcje często rozpoczynają się od słowa kluczowego, np. var, let, const, if, for, function.
- Przeglądarka wykonuje instrukcje JS w kolejności ich umieszczenia w kodzie.
- Dla lepszej czytelności kodu dobrze jest unikać długich linii oraz używać spacji (np. przy operatorach).
- Umieszczając zestaw instrukcji w nawiasach { } tworzymy blok kodu, który ma być wykonywany jako całość.

Komentarz

Komentarze w kodzie JS umieszczany są następująco:

- blokowo – pozwala wykomentować wiele linii

```
/* komentarz blokowy
zajmuje kilka linii */
```

- liniowo – obowiązuje tylko do końca jednej linii

```
// komentarz liniowy
```

JS – składnia

Dane

Dostępne są następujące typy danych: **typ liczbowy** (całkowity i rzeczywisty), **typ łańcuchowy**, **typ logiczny**, **typ obiektowy** i in.

Zmienne

- Zmienne (pojemniki na dane) deklarujemy przez użycie słów kluczowych let lub const. Jeśli wartość będzie zmiennej będzie zmieniana – użyj let, jeśli nie – użyj const.
- Zawsze deklaruj zmienne, koniecznie przed ich użyciem.
- Każda zmienna posiada unikalną nazwę, nazywaną identyfikatorem.
- Nazwa zmiennej zaczyna się od litery lub znaku _ lub $ i może zawierać dodatkowo liczby. Duże i małe litery są rozróżniane.
- Zmienne zadeklarowane poza funkcjami są globalne, czyli dostępne w całym skrypcie. Zmienne zadeklarowane wewnątrz funkcji są lokalne i można z nich korzystać tylko wewnątrz tej funkcji.

Przykłady:

```
let a = 100;
let c = b + a;
const b = 50;
```

~~`a = 100;`~~ (taka automatyczna deklaracja nie jest zalecana)

~~`var a = 100`~~ (deklaracja zmiennych do 2015, obecnie nie jest zalecana)

Na zmiennych można wykonywać następujące operacje: arytmetyczne, przypisania, porównywania, bitowe, logiczne i inne.

Więcej informacji o zmiennych: www.w3schools.com/js/js_variables.asp

JS – składnia

Operatory

W JS istnieją różne rodzaje operatorów:

- Operatory arytmetyczne – wykonują działania matematyczne (+, -, *, /, %).
- Operatory przypisania – przypisują wartości zmiennym (=, +=, -=).
- Operatory porównania – porównują wartości i zwracają true lub false (==, >, <, !=).
- Operatory logiczne – służą do tworzenia złożonych warunków (&&, ||, !).
- Operatory tekstowe – łączą napisy (np. +).

I wiele innych...

Przykład:

```
let a = 3;
let x = (100 + 50) * a;
```

```
let wynik = (x > y) && (y > 0);
```

Więcej na ten temat:

www.w3schools.com/js/js_operators.asp
www.w3schools.com/js/js_arithmetic.asp
https://www.w3schools.com/js/js_assignment.asp
https://www.w3schools.com/js/js_comparisons.asp

JS – składnia

Łańcuchy znaków (napisy, string)

String to typ danych służący do przechowywania tekstu.
Tekst można zapisywać w cudzysłowach (" ") lub apostrofach (' ,).
Przykład:

```
let imie = "Jan";
let nazwisko = "Kowalski";
let osoba = imie + " " + nazwisko;
```

Liczby (Numbers)

Typ Number służy do przechowywania liczb całkowitych i rzeczywistych.
JavaScript nie rozróżnia typów int i float – wszystkie liczby są typu Number.
Wszystkie liczby w JavaScript są przechowywane jako 64-bitowe liczby zmiennoprzecinkowe podwójnej precyzji (double precision floating point).

Instrukcje sterujące

JavaScript umożliwia wykorzystanie standardowych instrukcji sterujących takich jak:

- instrukcje warunkowe **if**, **if…else**, **if…else if**
- instrukcje wyboru: **switch…case**
- operator warunkowy: **? :**
- pętle: **for**, **for...in, for…of**, **while**, **do...while**

Więcej na: www.w3schools.com/js/js_strings.asp, www.w3schools.com/js/js_numbers.asp, www.w3schools.com/js/js_if.asp, www.w3schools.com/js/js_loops.asp

JS – składnia

Funkcje

W JavaScript możemy tworzyć własne funkcje (wydzielone bloki kodu przeznaczone do wykonywania konkretnych zadań) w następujący sposób:

```
function nazwa_funkcji(argument1, argument2, …)
{
instrukcje wnętrza funkcji
}
```

przykład (dodawanie dwóch liczb):

```
<script>
   function dodaj(a, b)
   {
      const wynik = a + b;
      return wynik;

   }
   const suma = dodaj (1, 2);
   document.write("Wynikiem dodawania 1 + 2 jest " + suma + ".");



</script>
```

przerwanie działania funkcji i zwrot wyniku, kod funkcji znajdujący się po return nie jest wykonywany

wywołanie funkcji z określonymi argumentami

instrukcja pozwalająca na umieszczanie tekstu na stronie www

Więcej: www.w3schools.com/js/js_functions.asp

JS – składnia

Tablice

Tablice służą do przechowywania wielu wartości w jednej zmiennej. Tablice są obiektami.

Elementy tablicy są zapisywane w nawiasach kwadratowych [] i oddzielane przecinkami.

Każdy element ma swój indeks. Numeracja zaczyna się od 0.

Tablice mają rozmiar dynamiczny – nie deklarujemy ich rozmiaru, a elementy można dodawać lub usuwać.

Mogą przechowywać dane różnych typów (liczbowe, łańcuchy, inne tablice, …)

Deklarację tablicy wykonuje się zwykle kluczem const. Nie oznacza to, że elementy tablicy muszą być stałe. Ich wartości można zmieniać. Można zmieniać liczbę elementów (dodawać, usuwać).

```
const kolory = ["czerwony", "zielony", "niebieski"];
```
deklaracja za pomocą literału tablicowego

```
kolor[0] = "różowy";
```
zmiana wartości pierwszego elementu

```
kolory.push("żółty");
```
wywołanie metody push dla obiektu będącego kolor (tablicy) – dodanie elementu

W starszych wersjach JavaScript tablicę deklarowało się kluczem var i konstuktorem Array, służącym do tworzenia nowych tablic:

```
var kolory = new Array("czerwony", "zielony", "niebieski");
var kolory = ["czerwony", "zielony", "niebieski"];
```

JS – składnia

Obiekty

JavaScript jest językiem obiektowym. Obiekty są zmiennymi, które mogą przechowywać zarówno wartości, jak i funkcje. Obiekty są jednym z najważniejszych pojęć w JavaScript, a ich zrozumienie jest niezbędne do opanowania podstawy działania JavaScript.

Właściwości obiektów (Object Properties):

- Obiekty JavaScript są zbiorami właściwości (properties).
- Właściwości można: zmieniać, dodawać, usuwać.

Metody obiektów (Object Methods):

- Metody to działania, które mogą być wykonywane przez obiekty.
- Metody są funkcjami przechowywanymi jako wartości właściwości obiektu.

Przykład:

```
const osoba = {
        imie: "Jan",
        przywitaj: function() {
        return "Witaj!";
        }
};
```

imie jest właściwością obiektu, przywitaj() jest metodą obiektu.

Więcej: www.w3schools.com/js/js_objects.asp

JS – składnia

Obiekty

- Obiekty są zmiennymi, które mogą przechowywać zarówno wartości, jak i funkcje.
- Wartości są przechowywane w postaci par klucz:wartość (key:value), nazywanych właściwościami (properties).
- Funkcje są przechowywane w postaci par klucz:funkcja (key:function()), nazywanych metodami (methods).

Obrazowe wyjaśnienie obiektów

Car Object


Car Properties	Car Methods
car.name = Fiat	car.start()
car.model = 500	car.drive()
car.weight = 850kg	car.brake()
car.color = white	car.stop()

Obiekt car:
Różne samochody mają te same właściwości, ale wartości tych właściwości mogą się różnić w zależności od samochodu.
Różne samochody mają również te same metody, ale mogą one być wykonywane w różnych momentach.

JS – składnia

Obiekty

Przykład tworzenia obiektu

```
const person = {
  firstName : "John",
  lastName : "Doe",
  age : 50,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

this oznacza odniesienie do tego obiektu (tu: obiektu person)

person.firstName – odwołanie do wartości właściwości firstName

person.fullName() – wywołanie funkcji (metody) fullName dla obiektu person

fullName – funkcja, która zwraca połączenie firstName, spacji i lastName

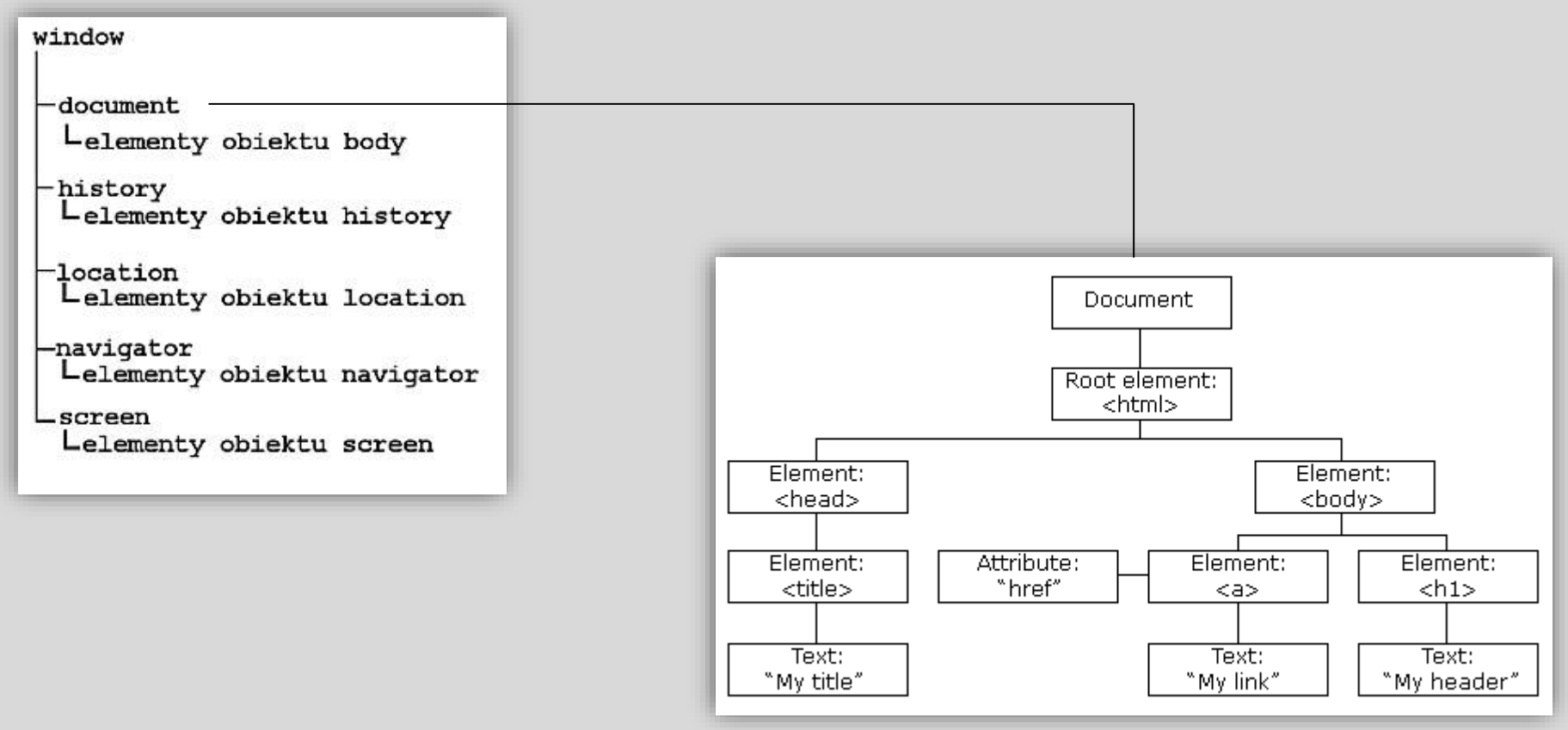
Wywołanie funkcji fullName i wpisanie tego co ona zwraca do dokumentu HTML. Wpisanie odbywa się z wykorzystaniem DOM (następny slajd):

```
document.getElementById("demo").innerHTML = person.fullName();
```

JS – DOM

obiektowy model dokumentu dla HTML

Obiektowy Model Dokumentu (DOM) HTML jest niezależnym od platformy i języka programowania interfejsem (API – Application Programming Interface), który umożliwia programom i skryptom (np. JavaScript) dynamiczny dostęp do zawartości dokumentu HTML, jego struktury i stylu oraz ich modyfikowanie. W modelu tym przeglądarka wraz z wyświetlana treścią jest zestawem obiektów. Obiekty w DOM ułożone są hierarchicznie (patrz obrazek). W modelu DOM elementy HTML są reprezentowane jako obiekty, dla których zdefiniowano właściwości, metody oraz zdarzenia.



JS – DOM

obiektowy model dokumentu dla HTML

DOM pozwala na dynamiczny dostęp i wprowadzanie zmian w zawartości, strukturze i stylu dokumentu. Dzięki temu, z poziomu JavaScript, elementy składające się na drzewo dokumentu HTML stają się obiektami na których możemy wykonywać szereg działań.

Wykorzystując DOM, JavaScript może:

- zmienić dowolny istniejący element HTML
- zmienić dowolny istniejący atrybut
- zmienić dowolną istniejącą deklarację CSS
- usunąć dowolny istniejący element HTML i atrybut
- dodać nowe elementy HTML lub atrybuty
- odpowiadać na dowolne zdarzenie HTML obecne na stronie
- tworzyć nowe zdarzenia HTML

JS – DOM

obiektowy model dokumentu dla HTML

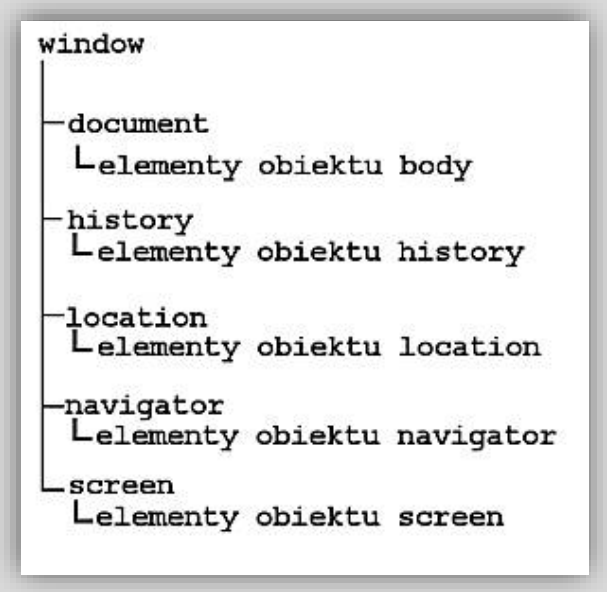
Obiekty DOM

- **window** – znajduje się na szczycie hierarchii i przedstawia okno lub kartę przeglądarki, gdzie wyświetlana jest strona WWW. Jest obiektem domyślnym i przy wywoływaniu jego metod i własności można stosować zapis skrócony.

  window.alert("*tekst*"); lub krócej alert("*tekst*"); — wyświetlanie okna dialogowego z *komunikatem*

  window.innerWidth — zwraca szerokość okna przeglądarki

  Możliwe są m.in. następujące operacje:
  wyświetlanie komunikatów / okien dialogowych
  otwieranie/zamykanie okna,
  zmiana rozmiarów i położenia okna,
  i in.



JS – DOM

Obiekty DOM, cd:

- **history** – zawiera historię stron odwiedzonych przez użytkownika podczas danej sesji przeglądarki (w danej karcie). Pozwala przechodzić do poprzednich i następnych stron w historii przeglądania, ale nie umożliwia odczytu adresów tych stron (ze względów bezpieczeństwa).
  - length – zwraca liczbę wpisów w historii bieżącej karty
  - back, forward – wczytuje poprzednią, następna stronę z historii
  - go(*parametr*) – wczytuje stronę wskazaną przez parametr. Liczba całkowita oznacza pozycję na liście historii odwiedzin (bieżący = 0, wstecz z minusem).

  Przykład:

  ```
  <button onclick="history.back()">Powrót</button>
  ```

  Wstawia przycisk po kliknięciu którego użytkownik zostanie przeniesiony do poprzednio odwiedzonej strony.
- **location** – zawiera informacje o adresie URL bieżącej wyświetlanej strony. Umożliwia odczytanie całości adresu i jego części składowy, np. nazwę protokołu (https, http, ftp). Przykładowe metody:
  - href – podaje pełny asders strony
  - protocol – podaje protokół
  - reload() – wymusza ponowne wczytanie bieżącej strony
  - assign(*url*) – wczytuje stronę podaną przez argument *url*, z zapisaniem bieżącej w historii
  - replace(*url*) – wczytuje stronę podaną przez argument *url*, bez zapisania bieżącej w historii (przekierowanie)

JS – DOM

Obiekty DOM, cd:

- **navigator** – zawiera informacje o przeglądarce internetowej i środowisku użytkownika: nazwa, i wersja przeglądarki, typ systemu operacyjnego, obsługa plików cookie połączenie z siecią i in.
  - navigator.userAgent – zawiera informacje o przeglądarce i systemie operacyjnym
  - navigator.language – informacja o języku przeglądarki
  - navigator.cookieEnabled – informacja, czy obsługa plików cookie jest włączona
  - navigator.online – informacja, czy przeglądarka ma połączenie z Internetem

JS – DOM

Obiekty DOM, cd:

- **document** – reprezentuje dokument HTML załadowany w przeglądarce. Umożliwia skryptom JavaScript dostęp do elementów strony oraz ich modyfikowanie. Najczęściej stosowane) metody obiektu document to:
  - getElementById(*id*) – zwraca informacje o elemencie HTML posiadającym zadane id (kod elementu i jego styl lokalny). Umożliwia dostęp do różnych części kodu HTML i np. zmianę ich zawartości.

    Przykład:

```
<body>

  <p id="demo">   </p>

  <script>
     // znajdź akapit z określonym id i przypisz go do zmiennej myPara
   const myPara = document.getElementById("demo");
     //zmień wartość właściwości innerHTML na podaną
   myPara.innerHTML = "Hello World!";
  </script>

</body>
```

    Efektem działania tego skryptu będzie wyświetlenie przez przeglądarkę akapitu z tekstem Hello World!

Możliwe jest też wyszukiwanie elementów po nazwie znacznika HTML lub nazwie klasy, i inne.

JS – DOM

Obiekty DOM, cd:

- **document** – najczęściej stosowane) właściwości/metody obiektu:
  - innerHTML – pozwala odczytać lub zmienić zawartość elementów HTML, w tym kodu HTML (przykład na poprzednim slajdzie)
  - textContent – podobniej jej wyżej, ale dotyczy tylko tekstu, który jest zawartością elementu HTML
  - write(*tekst*) – umieszcza w dokumencie zadany *tekst*. Jeśli zostanie wywołana po wczytaniu strony, nadpisze ją.

    Przykład:

```
<script>
  document.write("<p  id="akapit">");
  document.write("To jest treść napisana " + 22 + " maja");
  document.write("</p>");
</script>
```

Więcej o DOM: www.w3schools.com/js/js_htmldom.asp

JS – obsługa zdarzeń

Zdarzenia to akcje wykonywane przez użytkownika lub przeglądarkę. JavaScript może reagować na zdarzenia i wykonywać odpowiedni kod wpływający na wyświetlanie treści dokumentu. Przykłady zdarzeń: kliknięcie przycisku (onclick), najechanie myszą (onmouseover), załadowanie strony (onload), naciśnięcie klawisza (onkeydown).

Procedurę obsługującą zdarzenia wprowadzamy do kodu HTML w postaci atrybutu wybranego elementu:

```
<element zdarzenie="instrukcja">
<element zdarzenie="nazwa_funkcji()">
```

Poprawne jest użycie ” ” lub ’ ’.

Przykład:

```
<button onclick="document.getElementById('demo').innerHTML=Date()">podaj datę i godz.</button>

<p id="demo"></p>
```

po kliknięciu przycisku skrypt wypisze aktualną datę i godzinę

Więcej na temat obsługi zdarzeń: www.w3schools.com/js/js_events.asp

JS – automat do zapisywania LM

Poniżej zamieszczam kod umożliwiający automatyczne wpisanie jasności LM po podaniu liczby gwiazd N.

```
<td>                                                    komórka tabeli, w której użytkownik wybiera odpowiednią liczbę N
  <select name="N-pole3" id="pole3" size="1">
    <option value="5">5</option>
    <option value="6">6</option>
    <option value="7">7</option>
    <option value="8">8</option>
    <option value="9">9</option>
    <option value="11">11</option>
    <option value="13">13</option>
    <option value="14">14</option>
    <option value="15">15</option>
    <option value="16">16</option>
    <option value="17">17</option>
    <option value="18">18</option>
    <option value="19">19</option>
    <option value="20">20</option>
    <option value="23">23</option>
    <option value="25">25</option>
    <option value="27">27</option>
    <option value="29">29</option>
    <option value="33">33</option>
    <option value="37">37</option>
    <option value="44">44</option>
    <option value="49">49</option>
    <option value="54">54</option>
  </select>
  <button onclick="NdoLMpole3()">OK</button>        komórka zawiera też przycisk, którym użytkownik zatwierdza wybór, co powoduje wywołanie funkcji napisanej w JS
 </td>

 <td id="wynpole3"> </td>                           komórka, do której ma być wpisana wartość LM – w efekcie działania skryptu JS
```

Ciąg dalszy na następnej stronie.

JS – automat do zapisywania LM

Blok <script> z funkcją wywoływaną po kliknięciu przycisku w tabeli.

```
<script>
   function NdoLMpole3() {
      const inpN = Number(document.getElementById("pole3").value);
      const Npole3 = [5, 6, 7, 8, 9, 11, 13, 14, 15, 16, 17, 18, 19, 20, 23, 25, 27, 29, 33, 37, 44, 49, 54];
      const LMpole3 = [4.5, 4.6, 4.8, 5.2, 5.4, 5.7, 5.8, 6.0, 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8, 6.9, 7.0, 7.1, 7.2, 7.3, 7.4, 7.5];
      const indN = Npole3.indexOf(inpN);
      document.getElementById("wynpole3").innerHTML = LMpole3[indN];
   }
</script>
```

funkcja pobiera wartość N wybraną przez użytkownika a następnie znajduje odpowiadającą jej wartość LM i wpisuje ją w odpowiedniej komórce w tabeli

Podobny kod należy napisać dla pozostałych obszarów nieba znajdujących się w raporcie.

To jest jeden ze sposobów realizacji tego zadania. Innym jest wykorzystanie obiektu Map, który służy do przechowywania par klucz → wartość (www.w3schools.com/js/js_maps.asp).