JavaScript

Podstawy JavaScript

- JavaScript (JS) jest obiektowym językiem skryptowym
- JS jest powszechnie stosowany na stronach internetowych w celu rozszerzenia interakcji z użytkownikiem
- Skrypty JS są wykonywane przez przeglądarkę użytkownika

Podstawy JavaScript

- Klasycznym przykładem wykorzystania możliwości JS jest walidacja poprawności wprowadzonych do formularza danych
- Walidacja odbywa się on-line, w trakcie wpisywania danych przez użytkownika
- Dzięki temu zmniejsza się ilość przesyłanych danych i operacji wykonywanych po stronie serwera

Podstawy JavaScript – osadzanie kodu

- Kod JS można osadzać w dokumentach HTML na 2 sposoby:
  - bezpośrednio w dokumencie HTML pomiędzy znacznikami

```
<script type="text/javascript">
   //Treść skryptu
</script>
```

Podstawy JavaScript – osadzanie kodu

– w pliku zewnętrznym, z linkowaniem w dokumencie HTML

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

Podstawy JavaScript – osadzanie kodu

- Skrypt JS może zostać umieszczony w dowolnym miejscu dokumentu HTML, w sekcji `head` lub `body`
- Ze względu na poprawienie szybkości ładowania strony zaleca się umieszczanie skryptów na samych końcu sekcji `body`

Podstawy JavaScript – komentarze

- Sposób wstawiania komentarzy jest podobny do tego w PHP:
  - `/*komentarz*/` - w ten sposób wstawimy komentarz blokowy, wieloliniowy,
  - `//komentarz` - w ten sposób wstawimy komentarz jednoliniowy

Podstawy JavaScript – wypisywanie informacji

- Do wypisywania informacji za pomocą JS służą 2 funkcje:
  - `document.writeln("treść komunikatu")` – funkcja ta wstawia znak nowej linii na końcu wiersza,
  - `document.write("treść komunikatu")` – funkcja ta nie wstawia znaku nowej linii na końcu wiersza

  UWAGA: w wersji produkcyjnej nie używamy tych metod. Bezpieczniejszym rozwiązaniem jest zastosowanie `innerHTML()`.

Podstawy JavaScript – wypisywanie informacji

- W połączeniu z ww. funkcjami dostępne są znaki specjalne poprzedzane "\", tj.:
  - `\n` – znak nowej linii,
  - `\t` – znak tabulatora,
  - `\r` – znak powrotu do początku linii
  - `\"` – znak cudzysłowu

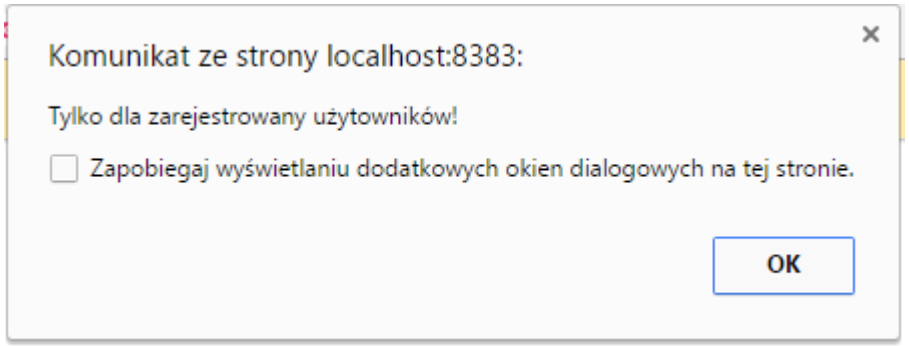
  UWAGA: znaki te (oprócz `\"`) nie są interpretowane przez przeglądarkę zgodnie z oczekiwaniami.

Podstawy JavaScript – wypisywanie informacji

- Inne sposoby wyświetlania komunikatów:
  - `window.alert("komunikat")` – wyświetla wyskakujące okienko z komunikatem oraz przyciskiem OK

Podstawy JavaScript – wypisywanie informacji

```
window.alert("Tylko dla
zarejestrowany użytowników!");
```

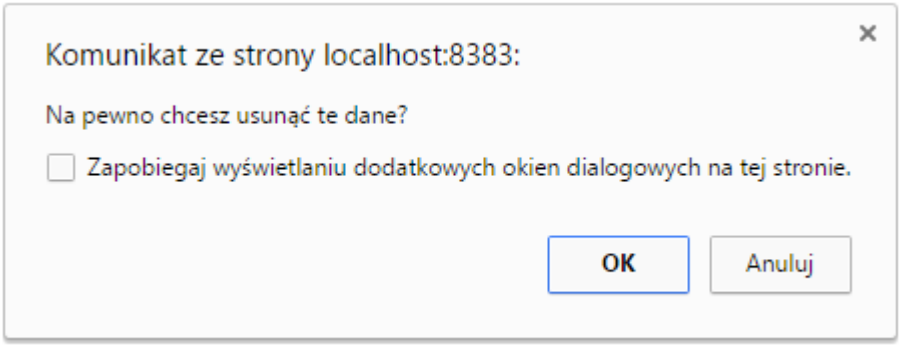


Podstawy JavaScript – wypisywanie informacji

- `window.confirm("komunikat")` – wyświetla wyskakujące okienko z komunikatem oraz przyciskami OK i Anuluj, zwraca `true` lub `false`

Podstawy JavaScript – wypisywanie informacji

```
window.confirm("Na pewno chcesz usunąć te dane?");
```

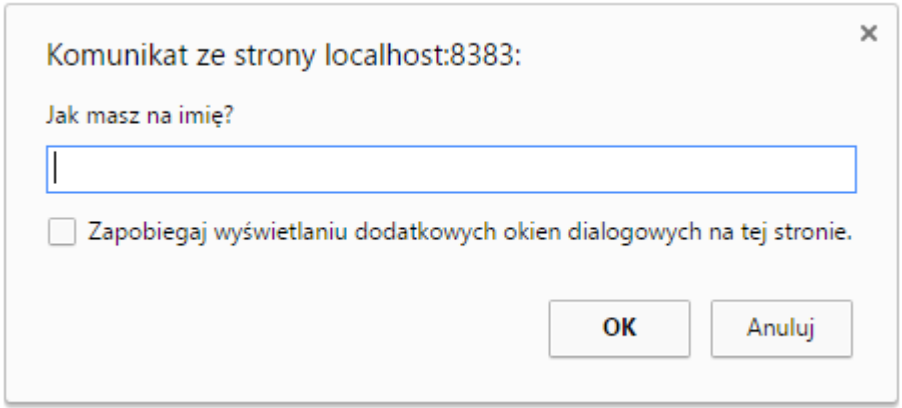


Podstawy JavaScript – wypisywanie informacji

- `window.prompt("komunikat")` – pozwala pobrać dane od użytkownika w celu dalszego przetwarzania

Podstawy JavaScript – wypisywanie informacji

```
window.prompt("Jak masz na imię?");
```



Podstawy JavaScript – zmienne

- Tak jak w innych językach programowania w JS można używać zmiennych do przechowywania porcji informacji
- Podobnie jak PHP, JS cechuje się słabym typowaniem zmiennych oraz łatwym konwertowaniem na inny typ

Podstawy JavaScript – zmienne

- Deklarowanie zmiennych w JS może mieć następującą postać:

```
var imie; //wartość null
imie = window.prompt("Jak masz na imię?");//to co wpisze użytkownik
var liczba = 10; // wartość 10
```

Podstawy JavaScript – zmienne

- Główne typy zmiennych w JS:
  - `number` – liczba całkowita lub zmiennoprzecinkowa, wielkość ograniczona możliwościami systemu operacyjnego
  - `string` – łańcuch znaków, umieszczany pomiędzy `""` lub `''`

Podstawy JavaScript – zmienne

  - `boolean` – typ logiczny, przechowuje wartość `true` lub `false`,
  - `null` – określenie braku wartości i typu
- JS rozpoznaje wielkość znaków, oznacza to, że `name` i `Name` to zupełnie inne zmienne
- Nazwa zmiennej może zaczynać się od litery, znaku `_` lub `$`

Podstawy JavaScript – konwersje

- W JS, odwrotnie niż w PHP, jeśli będziemy próbowali wykonywać dodawanie na zmiennych, z których jedna będzie ciągiem znaków, druga również zostanie zamieniona na ciąg znaków
- Dzieje się tak, ponieważ znak + wykorzystywany jest również do konkatenacji (łączenia) łańcuchów

Podstawy JavaScript – konwersje

```
document.writeln(6+"1");// 61
document.writeln(6+1);// 7
document.writeln(6+true);// 7
document.writeln(3*"2");// ???
document.writeln(7/false);// ???
```

Podstawy JavaScript – konwersje

- Aby skonwertować zmienną `string` na typ liczbowy, należy użyć funkcji `parseInt()` – konwersja na liczbę całkowitą, lub `parseFloat()` – konwersja na liczbę zmiennoprzecinkową

Podstawy JavaScript – konwersje

```
document.writeln(6.5+parseInt("1"));// 7.5
document.writeln(parseInt(6.5)+parseInt("1"));// 7
document.writeln(6+parseFloat(1));// 7
parseFloat(document.writeln(6+true));// 7
```

Podstawy JavaScript – konwersje

- W przypadku, gdy konwersja nie będzie możliwa (np. konwersja "abc" do liczby całkowitej), zostanie zwrócona wartość `NaN`, czyli `Not a Number`
- Aby obsłużyć tego typu sytuacje, można posłużyć się funkcją `isNaN()`, która zwraca wartość `true` lub `false` w zależności od argumentu

```
var liczbaA, liczbaB;
liczbaA = window.prompt("Podaj liczbę A");
liczbaB = window.prompt("Podaj liczbę B");
if (isNaN(liczbaA) || isNaN(liczbaB)){
    document.write("Jedna z wartości nie jest liczbą.");}
else {
    document.write(parseFloat(liczbaA) * parseFloat(liczbaB));}
```

Podstawy JavaScript – operatory

- JS udostępnia operatory znane z PHP:
  - = - przypisania wartości do zmiennej
    `var a = 7;`
  - arytmetyczne - `+`, `-`, `*`, `/`, `%` (modulo)
  - ciągów – `"cudzysłów"`, `'apostrof'`, `+` - konkatenacji

```
var string = "jakiś" + ' ' + 'tekst';
document.write(string);// jakiś tekst
```

Podstawy JavaScript – operatory

– łączone operatory - +=, –=, *=, /=, %=

```
var a = 10;
a += 7;     //<=> a = a + 7;
document.writeln(a);// 17
```

– operatory inkrementacji ++ i dekrementacji ––, przy czym możliwa jest zarówno pre- jak i post-inkrementacja/dekrementacja

Podstawy JavaScript – operatory

– operatory porównań:

- == - równość,
- === - dokładna równość,
- != - nierówność,
- < - mniejszość,
- > - większość,
- <= - mniejszość lub równość,
- >= - większość lub równość

Podstawy JavaScript – operatory

– operatory logiczne:

- && - oraz,
- || - lub,
- ! - negacja

Podstawy JavaScript – operatory

- Operatory logiczne i porównań wykorzystywane są, podobnie jak w PHP, do sterowania programem i wykonywania kodu w zależności od wprowadzonych przez użytkownika danych bądź otrzymanych wyników

Podstawy JavaScript – instrukcje warunkowe

- JS udostępnia (podobnie jak wszystkie inne języki programowania) instrukcje warunkowe `if`, `else`, `else if` oraz `switch`
- Należy zwrócić uwagę, że instrukcja `else if` zawiera odstęp pomiędzy wyrazami, w PHP wyrażenie `elseif` zapisujemy bez spacji

Podstawy JavaScript – instrukcje warunkowe

```
var imie = window.prompt("Jak masz na imię?")
if (imie == "Marcin"){
     document.write("Mam Cię!")
} else if (imie != ''){
     document.write("Witaj " + imie + "!")
} else {
     document.write("Witaj anonimie!")
}
```

Podstawy JavaScript – instrukcje warunkowe

```
var date = new Date()
var month = date.getMonth()
switch (data) {
      case 4:
        document.write("Gdy kwitną kasztany…")
        break
      default:
        document.write("Aby do maja…")
        break
}
```

Podstawy JavaScript – pętle

- JS udostępnia również znane z PHP pętle `while`, `do while` oraz `for`
- Ich konstrukcja i funkcjonowanie są tożsame z PHP

Podstawy JavaScript – pętle (while)

```
var liczba = 0
while (liczba < 10 ||
isNaN(liczba)){
     liczba = window.prompt("Podaj
liczbę większą od 10")
}
document.write("Podałeś liczbę " +
liczba)
```

Podstawy JavaScript – pętle (do while)

```
do {
    var liczba =
window.prompt("Podaj liczbę większą
od 10")
} while (liczba < 10 ||
isNaN(liczba))
document.write("Podałeś liczbę " +
liczba)
```

Podstawy JavaScript – pętle (for)

```
var num = 7;
for (var i = 1; i <= 9; i++){
    document.writeln(i + " * " +
num + " = " + i*num + "\n")
}
```

Podstawy JavaScript

Zadanie 1.pdf

Podstawy JavaScript – tablice

- Podobnie jak PHP, JS udostępnia 2 rodzaje tablic:
  - indeksowane numerycznie, np.:

```
var t = new Array("Coolpix", "K-1", "Agfa")
document.write(t[0])//Coolpix
```

Podstawy JavaScript – tablice

– asocjacyjne, indeksowane dowolnym kluczem, np.:

```
var t = {"Nikon": "Coolpix",
"Pentax": "K-1"}
document.write(t["Pentax"]) //K-1
```

Podstawy JavaScript – tablice

- W odróżnieniu od PHP, JS tablice są obiektami i mają swoje metody (funkcje)
- Aby wywołać metodę obiektu klasy `Array`, po nazwie obiektu i kropce podajemy nazwę metody, np.:
  `t.length` – zwraca długość (liczbę elementów) tablicy `t`

Podstawy JavaScript – tablice

- Aby dodać element na końcu tablicy, należy skorzystać z metody `push()`, np.

```
var t = ["owoce", "warzywa"];
t.push("nabiał");
for (var i in t) {
    document.writeln(t[i]);
}
```

Podstawy JavaScript – tablice

- Możemy również podać numer pozycji, pod którą ma zostać wstawiony element, np.:

```
t[3] = "mięso";
for (var i in t) {
    document.writeln(t[i]);
}
```

Podstawy JavaScript – tablice

- Do łatwego przetwarzania w pętli wszystkich elementów tablicy można użyć pętli `for … in`
- Działa inaczej niż `foreach` w PHP, nie tworzy nowych zmiennych a jedynie sama sprawdza ilość elementów w tablicy oraz ustawia licznik na 0 i zwiększa go o 1 po każdej iteracji

Podstawy JavaScript – tablice

```
var animals = new Array("koty",
"psy", "chomiki")
for (var i in animals){
    document.writeln(animals[i]+"
<br>")
}
```

Podstawy JavaScript - funkcje

- Funkcja to kod przeznaczony do wykonywania określonych zadań, który może być wykorzystywany wielokrotnie
- Dzięki funkcjom zmniejsza się ilość kodu
- Dodatkowo, w przypadku błędów lub zmiany koncepcji, poprawki wprowadzamy w jednym miejscu

Podstawy JavaScript - funkcje

- Podstawowa deklaracja funkcji ma następującą postać:

```
function nazwaFunkcji(){
    //kod do wykonania
}
```

Podstawy JavaScript - funkcje

- Dodatkowo funkcja może przyjmować argumenty/paramatry, czyli obiekty lub zmienne, które będą przetwarzane w funkcji:

```
function funkcjaZparametrami(par1, par2){
    //kod do wykonania
}
```

Podstawy JavaScript - funkcje

```
function writeGreeting(){
    document.writeln("Witaj!");
}

writeGreeting();//Witaj!
```

Podstawy JavaScript - funkcje

```
function writeGreeting(name){
    document.writeln("Witaj " + name
+"!");
}

imie = window.prompt("Jak masz na
imię?");
writeGreeting(imie);
```

Podstawy JavaScript - funkcje

```
function writeGreeting(name){
   if (name){
      document.writeln("Witaj " + name +"!");
   } else {
      document.writeln("A więc nie chcesz podać
   imienia?");
   }
}
writeGreeting(window.prompt("Jak masz na
imię?"))
```

Podstawy JavaScript - funkcje

```
function write(a){
      if (Array.isArray(a)){
            for (var i in a){
                document.writeln(a[i]+"<br>");
           }
      } else {
           document.writeln(a)
      }
}
```

Podstawy JavaScript - funkcje

- Oprócz wykonywania pewnych operacji, funkcje mogą również zwracać wynik
- Realizowane jest to za pomocą dyrektywy return

```
function add(a, b){
    return a+b;
}
```

Podstawy JavaScript - funkcje

```
var a = 7, b = 8;
function sum(a, b){
    return a+b;
}
function square(a){
    return a*a;
}
```

Podstawy JavaScript - funkcje

```
function squareSum(a, b){
    return
        sum(square(a), square(b));
}

write(squareSum(a, b));
//113
```

Podstawy JavaScript - funkcje

- Do tej pory deklaracje wszystkich zmiennych poprzedzaliśmy wyrażeniem `var`
- Ogólnie wyrażenie to nie jest konieczne do zadeklarowania zmiennej, ale ma bardzo duże znaczenie w przypadku funkcji, ponieważ określa zasięg zmiennej

Podstawy JavaScript - funkcje

- Jeśli w funkcji zadeklarujemy zmienną z użyciem wyrażenia `var`, to będzie miała ona zasięg lokalny – po wyjściu z funkcji nie będzie już dostępna
- Jeśli natomiast pominiemy wyrażenie `var`, to będzie miała zakres globalny – będzie dostępna poza ciałem funkcji

Podstawy JavaScript - funkcje

- Rodzi to jednak niebezpieczeństwo przesłonięcia wcześniej już zadeklarowanej zmiennej o tej samej nazwie
- Zmienne deklarowane poza ciałem funkcji zawsze mają zasięg globalny (po zadeklarowaniu są dostępne z dowolnego miejsca skryptu)

Podstawy JavaScript - funkcje

```
var l1 = 7, l2 = 9;
sum(l1, l2)
function sum(a, b){
    var wynik = a + b;
    return wynik;
}
document.writeln(wynik);// (nic)
```

Podstawy JavaScript - funkcje

```
var l1 = 7, l2 = 9;
sum(l1, l2)
function sum(a, b){
    wynik = a + b;
    return wynik;
}
document.writeln(wynik);//16
```

Podstawy JavaScript - funkcje

- Aby móc skorzystać z funkcji, musi ona zostać zadeklarowana pomiędzy tymi samymi znacznikami co jej wywołanie
- Możliwe jest również wywołanie funkcji z zewnętrznego pliku, przy czym plik musi zostać dołączony do dokumentu HTML przed wywołaniem funkcji

Podstawy JavaScript - funkcje

```
<script type="text/javascript"
src="js2.js"></script>
<script type="text/javascript>
    suma(7, 5);
</script>//12
//Funkcja suma jest zadeklarowana w
pliku js2.js
```

Podstawy JavaScript - funkcje

```
<script type="text/javascript">
function suma(a, b){
    document.write(a+b);}
</script>
<script type="text/javascript>
    suma(7, 5);
</script>// nie zadziała
```

Podstawy JavaScript - funkcje

```
<script type="text/javascript>
function suma(a, b){
    document.write(a+b);
}
suma(7, 5);
</script>//12
```

Podstawy JavaScript - funkcje

Zadanie 2.pdf

Obiekty

- W dużym uproszczeniu, obiekty są zmiennymi zawierającymi inne zmienne oraz posiadające metody przypisane tylko i wyłącznie do swojej klasy
- Wszystkie obiekty danej klasy mają taką samą strukturę

Obiekty

- Z pojęciem obiektu/klasy wiąże się również pojęcie instancji, czyli wystąpienia
- Instancja danej klasy to konkretny obiekt z konkretnymi wartościami właściwości
- Aby utworzyć obiekt danej klasy, należy użyć wyrażenia `new nazwaKlasy()`

Obiekty

```
function Employee(name, surname, email, pesel){
      this.name = name;//właściwość
      this.surname = surname;
      this.email = email;
      this.pesel = pesel;
      this.fullName = function(){
          return this.name + " " + this.surname;
      };//metoda
};//definicja klasy pracownik
```

Obiekty

```
var kowalski = new Employee("Jan",
"Kowalski", "kowalski@wp.pl",
"80121145966");//instancja klasy
Employee
document.write(
    kowalski.fullName());
document.write(kowalski.name);// ??
```

Document Object Model

- Document Object Model jest sposobem, w jaki JavaScript widzi dokument HTML
- DOM jest interfejsem niezależnym od platformy i języka programowania
- Pozwala on programom i skryptom dynamicznie modyfikować zawartość, styl i strukturę dokumentów HTML i XML

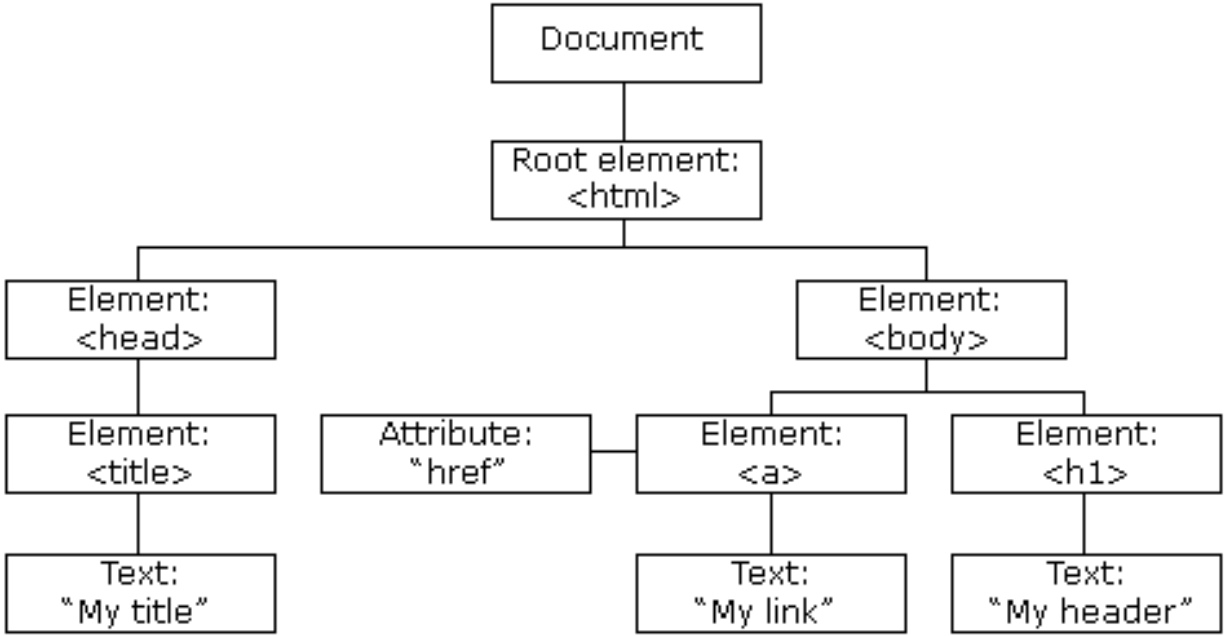
Document Object Model

- Standard DOM określony przez W3C jest podzielony na 3 części
  - Core DOM –model standardu dla wszystkich typów dokumentów
  - XML DOM – model standardu dla dokumentów XML
  - HTML DOM – model standardu dla dokumentów HTML

Document Object Model

- HTML DOM jest interfejsem i zorientowanym obiektowo modelem dla HTML. Definiuje:
  - elementy HTML jako obiekty
  - właściwości wszystkich elementów HTML
  - metody dostępowe do elementów HTML
  - zdarzenia dla elementów HTML

Document Object Model



źródło: https://www.w3schools.com/js/js_htmldom.asp

Document Object Model

- Każda strona wyświetlana w oknie przeglądarki staje się obiektem klasy Document
- Każdy element będący potomkiem obiektu Document również jest obiektem odpowiedniej klasy, posiadającym właściwości i metody

Document Object Model

- Obiekt Document udostępnia metody i właściwości pozwalające na dostęp do wszystkich elementów potomnych
- Metoda jest akcją, którą można wykonać na danym obiekcie, np. wyszukanie konkretnego elementu na stronie po jego id

```
document.getElementById("id")
```

https://www.w3schools.com/js/js_htmldom_document.asp

Document Object Model

- Metodą będzie wypisanie w oknie przeglądarki jakiejś treści

```
document.write("treść")
```

- Również wyświetlenie okna powiadomienia jest metodą wywoływaną z obiektu klasy Window

```
window.alert("komunikat")
```

Document Object Model

- Właściwość (property) jest wartością, którą można pobrać lub ustawić, np. zawartość elementu `id="help"`

```
document.getElementById("help").innerHTML - pobranie wartości


document.getElementById("help").innerHTML = "nowa treść" - ustawienie wartości
```

Document Object Model

- Najczęściej wykorzystywane metody obiektu Document
  - `getElementById("id")` – wyszukuje element o konkretnym id

```
<div id='welcome'>Witamy!</div>
document.getElementById('welcome');
   //zwraca obiekt o id 'welcome'
```

Document Object Model

- `getElementsByTagName("tag")` – zwraca wszystkie elementy będące określonymi znacznikami

```
document.getElementsByTagName("div");
```

- `getElementsByClassName("class_name")` – zwraca wszystkie elementy będące określonymi znacznikami

```
document.getElementsByClassName("red");
```

Document Object Model

- `querySelectorAll("selector")` – zwraca wszystkie elementy zdefiniowane za pomocą selektora CSS

```
document.querySelectorAll("div.red");
```

Document Object Model

- Najczęściej ustawiane/pobierane właściwości elementów:
  - `innerHTML` – zawartość HTML elementu

```
document.getElementById("help").innerHTML
```

https://www.w3schools.com/js/js_htmldom_html.asp

Document Object Model

– dowolny_atrybut – ustawienie pobranie dowolnego atrybutu

```
document.getElementById('link').href;
document.getElementById('link').href = nowa_wart;

document.getElementById('img').src;
document.getElementById('img').src = nowa_wart;
```

Document Object Model

– style.właściwość_css – ustawienie/pobranie dowolnej właściwości CSS

```
document.getElementById('img').style.width;
document.getElementById('img').style.color = nowa_wart;
```

Obiekty JS

- Inne klasy dostępne w JS oraz ich wybrane metody i właściwości
  - String – ciąg znaków
    - `charAt()` – zwraca znak na podanej pozycji
    - `concat()` – zwraca kopię dwóch lub więcej połączonych łańcuchów
    - `indexOf ()` – zwraca pierwszą pozycję szukanego podciągu

https://www.w3schools.com/js/js_object_definition.asp

Obiekty JS

- `toLowerCase()` – zamienia wszystkie znaki w łańcuchu na małe
- `toUpperCase()` – zamienia wszystkie znaki w łańcuch na duże

```
text = "Mam na imię Marcin";
document.write(text.toUpperCase());
```

Obiekty JS

- Date – data
  - `getDate()` – zwraca numer dnia miesiąca (1-31)
  - `getDay()` – zwraca numer dnia tygodnia (0-6)
  - `getMonth()` – zwraca numer miesiąca (0-11)

```
data = new Date();
dzien = data.getDay();
document.write(dzien);
```

Obiekty JS

- `getFullYear()` – zwraca rok w postaci czterocyfrowej
- `getHours()` – zwraca godzinę (0-23)

– Array – tablica zainicjowana jako new Array()

- `toString()` – łączy wszystkie elementy przecinkiem i zwraca jako ciąg
- `join()` – podobnie jak toString(), przy czym mamy możliwość podania znaku pełniącego funkcję łącznika
- `pop()` – usuwa ostatni element z tablicy

Obiekty JS

- `push()` – wstawia element na koniec tablicy
- `indexOf()` – zwraca indeks elementu w tablicy

– Math – klasa Math jest klasą globalną, przy tworzeniu obiektów tej klasy nie używamy operatora new
  - `max(l1, l2, …)` – zwraca największą z podanych liczb
  - `pow(podst, pot)` – podnosi podstawę do określonej potęgi

Obiekty JS

- `random()` – zwraca liczbę pseudolosową w zakresie 0-1
- `round()` – zaokrągla liczbę do najbliższej liczby całkowitej
- `PI` –zwraca wartość PI

```
a = 5.75;
potega = Math.pow(a, 3);// ??
zaokr = Math.round(potega);// ??
document.write(Math.round(Math.random()*100));
```

Obiekty przeglądarki

- Window – reprezentuje okno przeglądarki, jest obiektem nadrzędnym. Pozostałe obiekty są obiektami potomnymi i przy ich wywoływaniu nie ma konieczności odwoływania się do klasy Window

```
window.alert(komunikat) <=> alert(komunikat)
window.document.getElementById() <=>
document.getElementById()
```

https://www.w3schools.com/js/js_window.asp

Obiekty przeglądarki

- Wybrane metody dla obiektu Window
  - `alert()/confirm()/prompt()` – wyświetla stosowne powiadomienie w małym okienku
  - `close()` – zamyka okno
  - `open()` – otwiera nowe okno
  - `scrollBy()` – przesuwa stronę o wskazaną liczbę pikseli

```
newWindow = open('http://onet.pl', 'Onet');
```

Obiekty przeglądarki

- Navigator – reprezentuje przeglądarkę
  - userAgent– zwraca informację o nagłówku wysyłanym przez przeglądarkę
  - language – zwraca ustawiony język przeglądarki
  - cookieEnabled – zwraca informację, czy włączona jest obsługa cookies

```
document.write(navigator.userAgent);
document.write(navigator.language);
```

https://www.w3schools.com/js/js_window_navigator.asp

Obiekty przeglądarki

- Location – pozwala pobrać bieżący adres strony
  - location.href – zwraca pełny adres strony
  - location.hostname – zwraca główny adres strony

```
document.write(location.href);
document.write(location.hostname);
//location.href = "http://google.pl";
```

https://www.w3schools.com/js/js_window_location.asp

Obiekty przeglądarki

- Screen – reprezentuje ekran na którym wyświetlana jest strona
  - screen.width – szerokość ekranu
  - screen.height – wysokość ekranu

```
document.write(screen.width);
if (screen.width < 640){
     location.href("http://m.telepolis.pl");
}
```

https://www.w3schools.com/js/js_window_screen.asp

Obiekty przeglądarki

- History – reprezentuje historię przeglądarki. Ze względów bezpieczeństwa JS daje jedynie możliwość przekierowania wstecz i do przodu (jeśli istnieje historia) względem aktualnie wyświetlanej strony
  - back() – przekierowanie wstecz
  - forward() – przekierowanie do przodu

```
history.back();
```

https://www.w3schools.com/js/js_window_history.asp

Zdarzenia

- Zdarzenia to akcje wywoływane przez użytkownika strony
- Zdarzeniem może być
  - zmiana wartości pola formularza lub menu rozwijanego (onchange)
  - wybranie pola formularza (onselect)
  - opuszczenie pola formularza (onblur)

https://www.w3schools.com/js/js_events_examples.asp

Zdarzenia

– najechanie myszką na element (onmouseover)
– opuszczenie pola elementu (onmouseout)
– kliknięcie myszką (onclik)
– podwójne kliknięcie myszką (ondblclick)
– wczytanie elementu (onload)

Zdarzenia

- Obsługę zdarzeń można dodać na 2 sposoby
  - bezpośrednio w znaczniku HTML

```
<h1 onclick="this.innerHTML =
'Dzięki!'">Kliknij mnie</h1>
```

Słowo this określa właśnie ten konkretny element.

Zdarzenia

```
<h1 onclick="changeText(this)">
Kliknij mnie</h1>

<script>
function changeText(obiekt) {
    obiekt.innerHTML = "Dzięki";
}
</script>
```

```
<div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="background-color:#D94A38; width:120px; height:20px;
padding:40px;">
Najedź myszką</div>

<script>
function mOver(obj) {
    obj.innerHTML = "Dzięki"
}

function mOut(obj) {
    obj.innerHTML = "Najedź myszką"
}
</script>
```

```
<div onmousedown="mDown(this)" onmouseup="mUp(this)"
style="background-color:#D94A38; width:90px; height:20px;
padding:40px;">
Kliknij i przytrzymaj</div>

<script>
function mDown(obj) {
    obj.style.backgroundColor = "#1ec5e5";
    obj.innerHTML = "Puść";
}

function mUp(obj) {
    obj.style.backgroundColor="#D94A38";
    obj.innerHTML=" Kliknij i przytrzymaj ";
}
</script>
```

Zdarzenia

- Jako Event Listener (zazwyczaj w oddzielnym pliku js)
  - `element.addEventListener(event, function);`

```
document.getElementById("h3").addEventListener(
"click", function (){
      alert("click");
      };
);//UWAGA: nie używamy przedrostka on (click
zamiast onclick, select zamiast onselect)
```

https://www.w3schools.com/js/js_htmldom_eventlistener.asp

Zdarzenia

```
<button id="myBtn">Kliknij</button>

<script type="text/javascript">
document.getElementById("myBtn").addEventL
istener("click", function(){
    this.style.backgroundColor = "red";
});
</script>
```

JavaScript

Zadanie 3.pdf

Pytania?

?

Dziękuję za uwagę!