JavaScript

tutorial version 1.31

Cechy języka

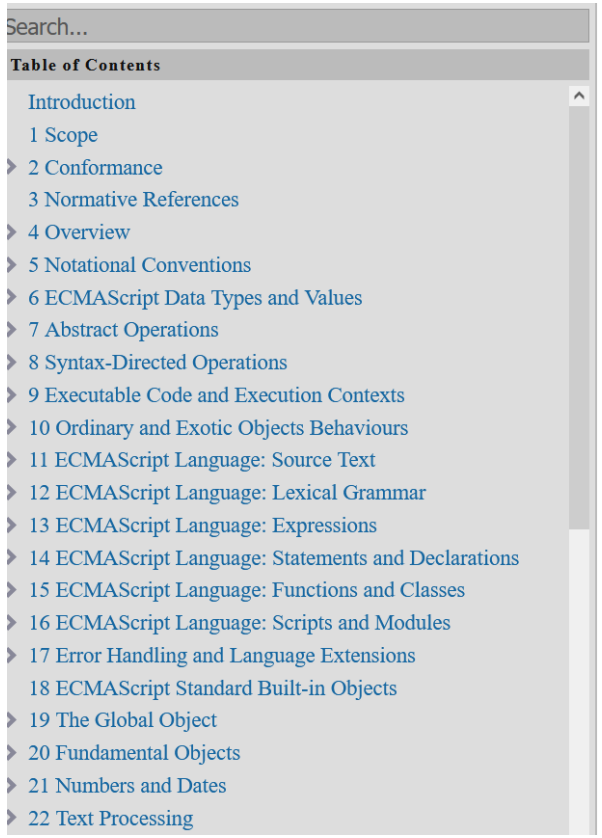
- obiektowy język skryptowy
- składnia i struktura języka podobna do C++ !!
- język interpretowany przez przeglądarkę
  (instrukcje są wykonywane bez wcześniejszej kompilacji
  - brak pliku wykonywalnego *.exe)
- skrypty są najczęściej umieszczane bezpośrednio
  na stronie internetowej (można je również
  ładować z innych komputerów, plików zewnętrznych)
- dodaje interaktywność do stron internetowych
- każdy może używać języka za darmo
  (nie trzeba kupować licencji)
- Java i JavaScript to nie to samo! Java jest językiem
  programowania o potężnych możliwościach
  (tak jak np. C++), JavaScript wygląda przy Javie skromnie.

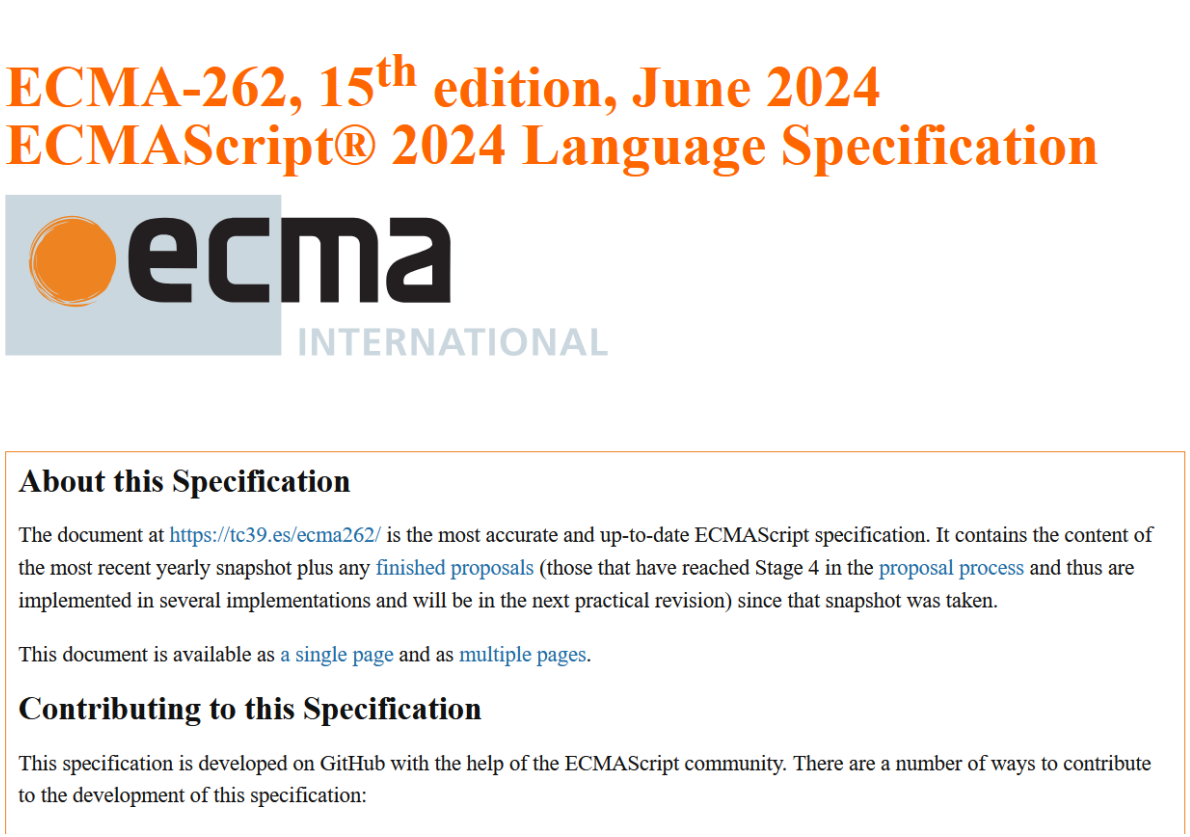
Możliwości

- JavaScript może umieszczać na stronie tekst zmieniający się dynamicznie
- JavaScript obsługuje zdarzenia – np. można napisać kod reagujący na kliknięcie myszą w dany element
- JavaScript może zmieniać treść elementów HTML na stronie
- JavaScript można wykorzystać do walidacji danych, czyli sprawdzenia ich poprawności przed wysłaniem na serwer
- JavaScript może rozpoznać typ przeglądarki, co można wykorzystać do załadowania odpowiedniej strony (w zależności od przeglądarki)
- JavaScript ma dostęp do plików cookies

ECMAScript

stworzona przez ECMA ustandaryzowana specyfikacja obiektowego języka programowania, której najbardziej znane implementacje to JavaScript, JScript i ActionScript.





https://pl.wikipedia.org/wiki/ECMAScript

Ciekawostki

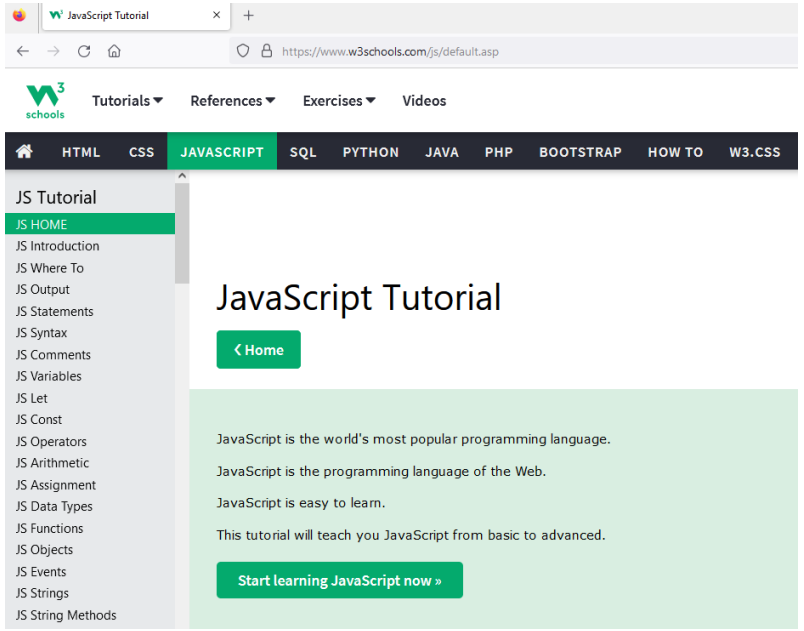
ECMA-262 jest oficjalnym standardem języka JavaScript.
Standard jest oparty na językach JavaScript (Netscape) oraz JScript (Microsoft).
Głównym twórcą języka jest amerykański programista

Brendan Eich

Ciekawostki

- Język po raz pierwszy pojawił się w przeglądarce Netscape Navigator 2.0 i od roku 1996 stał się dostępny we wszystkich przeglądarkach napisanych przez firmy Netscape i Microsoft
- W roku 1998 standard ECMA został zatwierdzony przez Międzynarodową Organizację Standaryzacyjną ISO (ISO/IEC 16262).
- Język ciągle się rozwija.

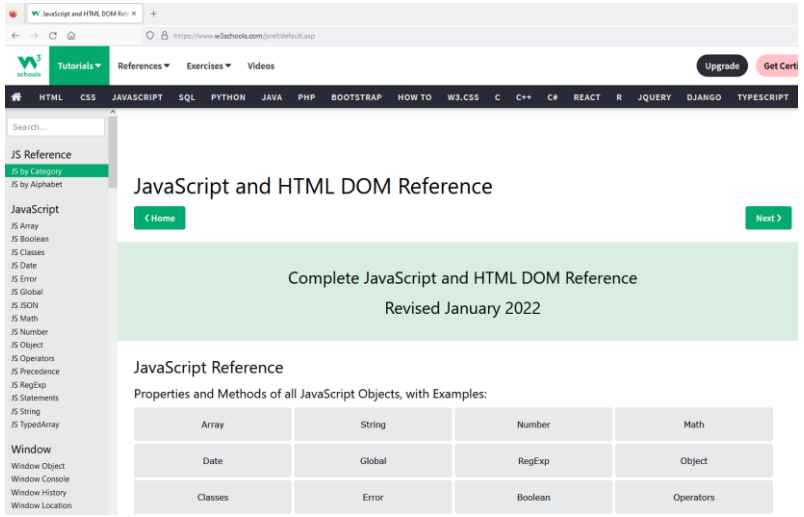
Tutorial



[js tutorial w3schools]

[mdn js]

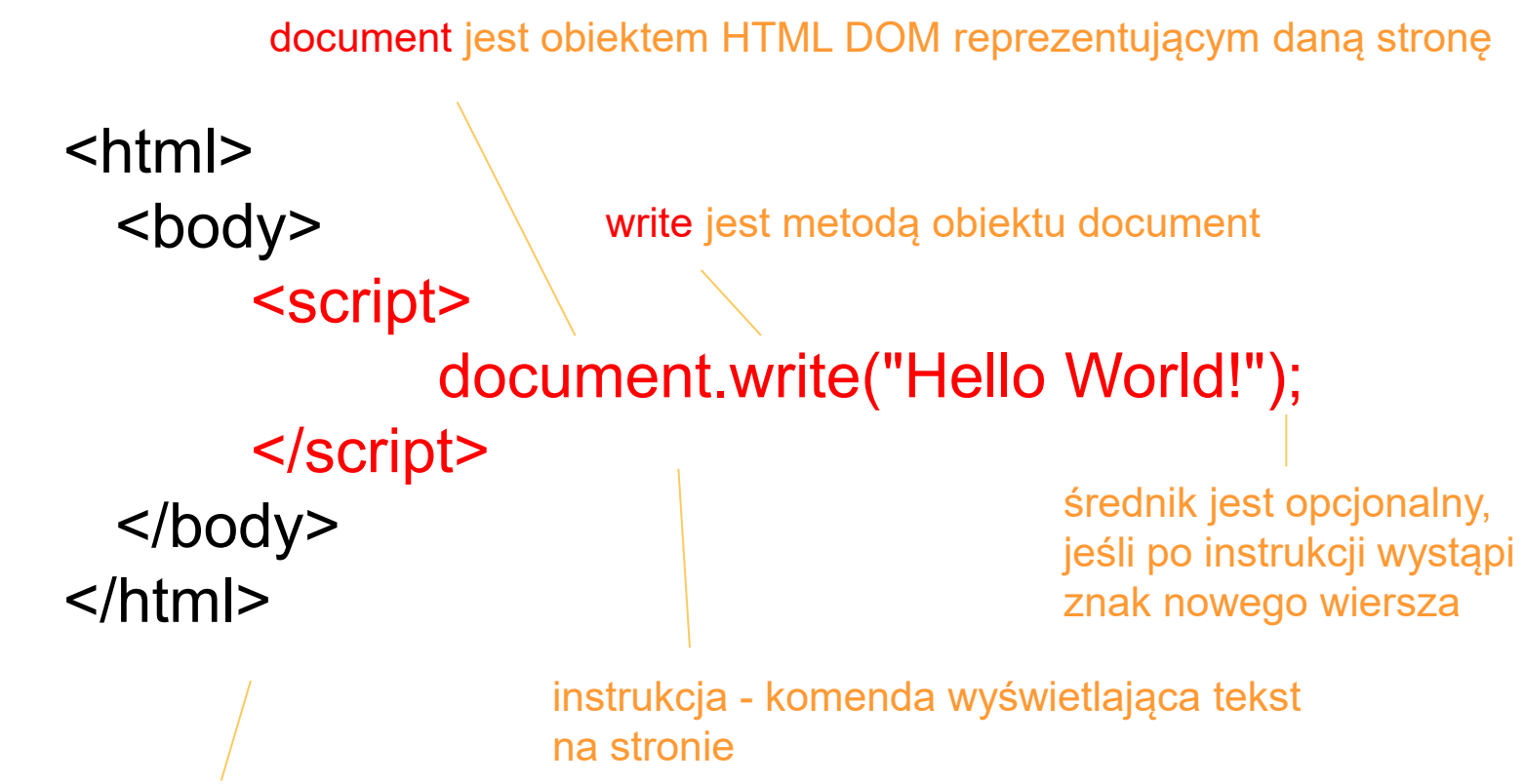
[https://javascript.info/first-steps]



[js reference w3schools]

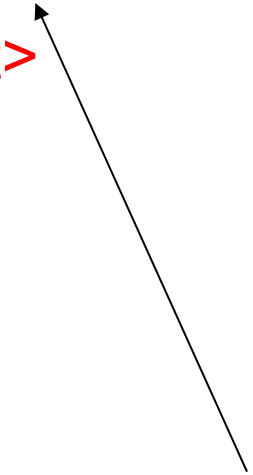
[js reference mdn]

Umieszczanie skryptu w kodzie strony w sekcji <body>



Umieszczanie skryptu w kodzie strony w sekcji <head>

```
<html>
  <head>
      <script>
      ……
      </script>
  </head>
  <body>
  </body>
</html>
```



kod skryptu umieszczony w sekcji <head>,
a wiec przed renderowaniem strony – dlatego w skrypcie
może nie być dostępu do elementów strony

Umieszczanie skryptu w kodzie strony w sekcji <head>

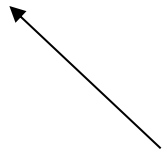
słowo kluczowe defer powoduje wykonanie skryptu po załadowaniu strony – a wiec elementy strony będą w nim dostępne

```
<html>
  <head>
      <script src="main.js" defer></script>
  </head>
  <body>
  </body>
</html>
```

kod skryptu umieszczony w sekcji <head> w zewnętrznym pliku main.js

Umieszczanie skryptu w zewnętrznym pliku

```
<html>
  <head>
      <script src=”skrypt.js”> </script>
  </head>
  <body>
  </body>
</html>
```

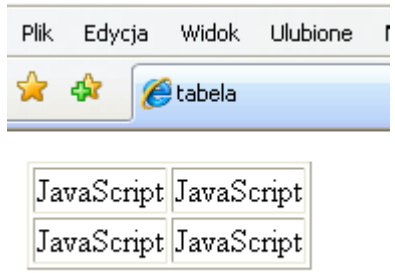


w zewnętrznym pliku
brak znaczników <script>

```
<html>
  <head>
      <script src=”http://www.plopl.com/skrypt.js”> </script>
  </head>
  <body>
  </body>
</html>
```

Ćwiczenia

1.1 Umieść w kodzie strony w sekcji <body> skrypt wyświetlający tabelę 2x2.
W komórkach tabeli
umieść słowo *JavaScript.*



1.2 Umieść w kodzie strony w sekcji <body> skrypt wyświetlający zdjęcie z wieżą Eiffla
(zjęcie może być inne)

Odpowiedzi do ćwiczeń

1.1

można napisać kod w jednej linijce

```
<html>
<head>

<title>tabela</title>
</head>
<body>

<script>
document.write("<table border=\"1\"><tr><td>JavaScript</td><td>JavaScript</td></tr><tr><td>JavaScript</td><td>JavaScript</td></tr></table>");
</script>

</body>
</html>
```

aby przeglądarka nie myliła znaku ” z końcem stringu
trzeba wstawić znak specjalny \” (czyli \”1\”)
(można również ..”.. ’1’.. ”..)

Odpowiedzi do ćwiczeń

1.1

```
<html>
<head>

<title>tabela</title>
</head>
<body>

<script type="text/javascript">
document.write("<table border=\"1\">\
                    <tr>\
                        <td>JavaScript</td>\
                        <td>JavaScript</td>\
                    </tr>\
                    <tr>\
                        <td>JavaScript</td>\
                        <td>JavaScript</td>\
                    </tr>\
                </table>");
</script>

</body>
</html>
```

aby napisać kod w wielu linijkach należy na końcu każdej z nich wstawić znak ukośnika

Odpowiedzi do ćwiczeń

1.2


```
<html>
<head>
<title>Wieża Eiffla</title>
</head>
<body>

<script>
document.write("<img src=\"wieza_eiffla.jpg\" alt=\"Wieża Eiffla\" />");
</script>

</body>
</html>
```

słowa kluczowe

słowa kluczowe, czyli wyrazy zarezerwowane
przez konstrukcję języka,
nie mogą być nazwami zmiennych

JavaScript Reserved Words

‹ Previous

Next ›

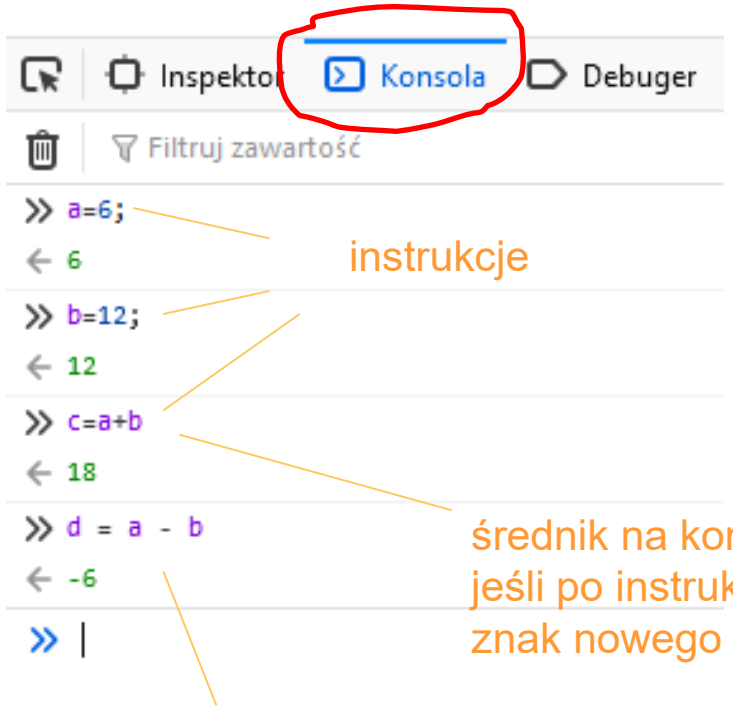
In JavaScript you cannot use these reserved words as variables, labels, or function names:

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

instrukcje Statements

script w js składa się z instrukcji,
czyli komend do wykonania przez przeglądarkę

Narzędzia Developerskie
F12



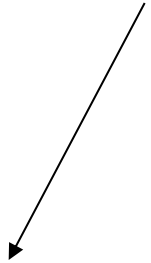
instrukcje

średnik na końcu instrukcji jest opcjonalny,
jeśli po instrukcji wystąpi
znak nowego wiersza

JavaScript ignoruje spacje

Ważne!

JavaScript is Case Sensitive



wielkość liter ma znaczenie!
zmienna *liczba* to nie to samo co zmienna *Liczba*

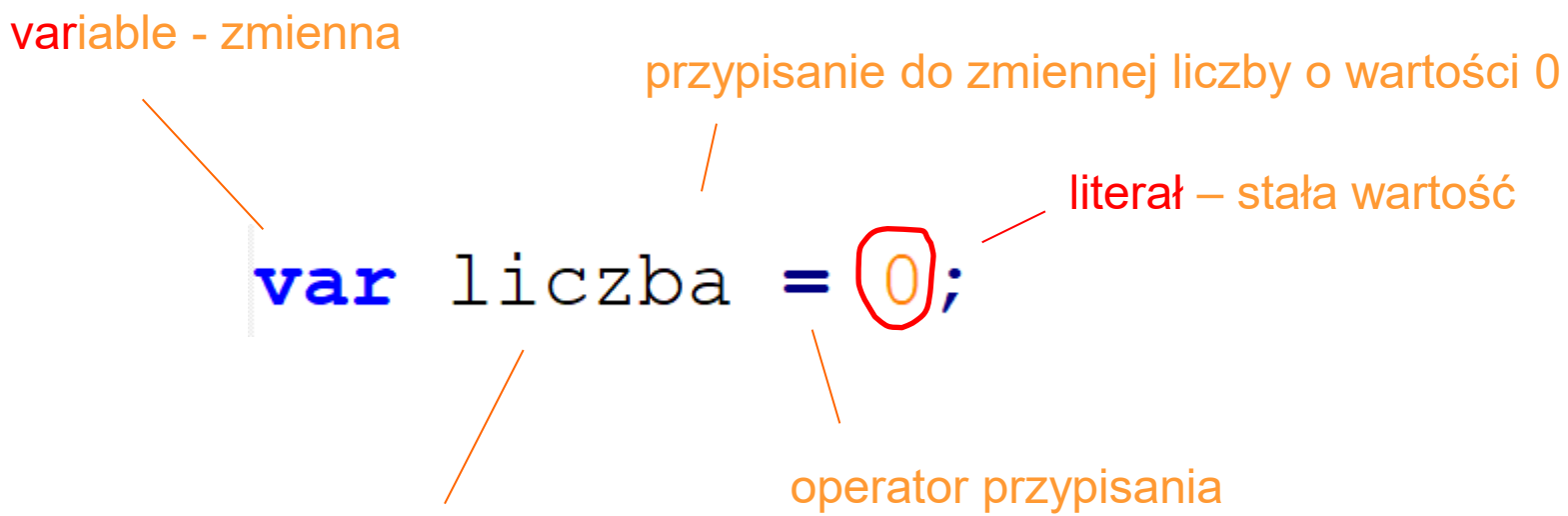
Komentarze

```
<script>
    //komentarz jednowierszowy pierwszy
    document.write("<h1>Oto moja strona</h1>");
    //komentarz jednowierszowy drugi
    document.write("<p>Chodzę do Liceum PŁ</p>");
    document.write("<p>Tutaj jest super!</p>");
    /*
    komentarz blokowy
    document.write("<p>Chodzę do Liceum PŁ</p>");
    document.write("<p>Tutaj jest super!</p>");
    */
</script>
```

komentarz jest ignorowany przez interpreter kodu
służy jako informacja dla programisty

Deklaracja i inicjalizacja zmiennych

zmienne służą do przechowywania danych



nazwa zmiennej:

- można stosować litery, cyfry, znak podkreślenia, znak dolara
- nazwa nie może zaczynać się od cyfry
- wielkość liter ma znaczenie
- nazwą nie może być słowo kluczowe JavaScript

Deklaracja i inicjalizacja zmiennych

typowanie dynamiczne – typ zmiennej jest dobierany w zależności od kontekstu (brak określenia typu zmiennej jak w C++)

jeśli do zmiennej zadeklarowanej z var nic nie przypiszemy zmienne mają wartość ”undefined”

```
// deklaracja zmiennych a, b oraz liczba
var a, b, liczba;

// przypisanie zera do zmiennej a
a = 0;

// deklaracja z przypisaniem wartości do zmiennej
let c = 0;

// deklaracja z przypisaniem wartości do zmiennej
const d = 0;
```

var było używane w js dawniej

średnik na końcu instrukcji jest opcjonalny (jeśli jest jedna instrukcja jest w jednej linii)

zmienna globalna jest widoczna w całym skrypcie, w którym została zadeklarowana

zmienna bez słowa kluczowego – zmienna globalna

let i const wprowadzono do języka w 2015

zmiennej d nie można zmienić – jest stałą

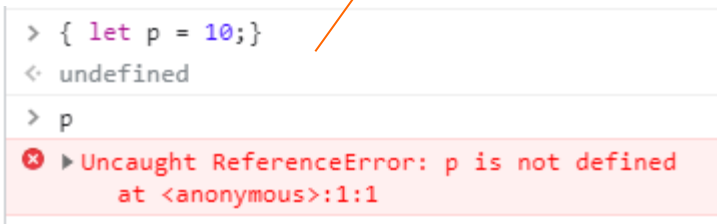
```
// można też podawać samą nazwę zmiennej przy jej inicjalizacji
zmienna = 40;
```

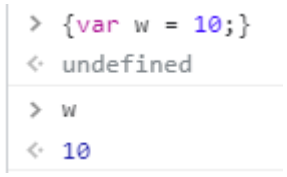
Deklaracja i inicjalizacja zmiennych

const i let mają zasięg bloku w którym zostały zadeklarowane

```
// deklaracja zmiennych w bloku
{
  let c = 0;
  const d= 0;
  var e = 0;
}
// zmienne c oraz d są poza blokiem niewidoczne
// zmienna e jest poza blokiem widoczna
```

zasięg zmiennych można przetestować w konsoli (F12)





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

zmienne lokalne

zasięg zmiennych można przetestować w konsoli (F12)

```
// zmienne imie, nazwisko, klasa są niewidoczne poza funkcją

function osoba() {

    let imie = „Anna";
    const nazwisko = „Kowalska”;
    var klasa = „Ib”;

    // zmienne imie, nazwisko, klasa są tutaj widoczne
    // (pomiędzy nawiasami klamrowymi)

}

// zmienne imie, nazwisko, klasa są niewidoczne poza funkcją
```

zmienne lokalne funkcji osoba()
czyli imie, nazwisko ,klasa
mają zasięg widoczności ograniczony
do wnętrza funkcji osoba()

zmienne ze słowami kluczowymi

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

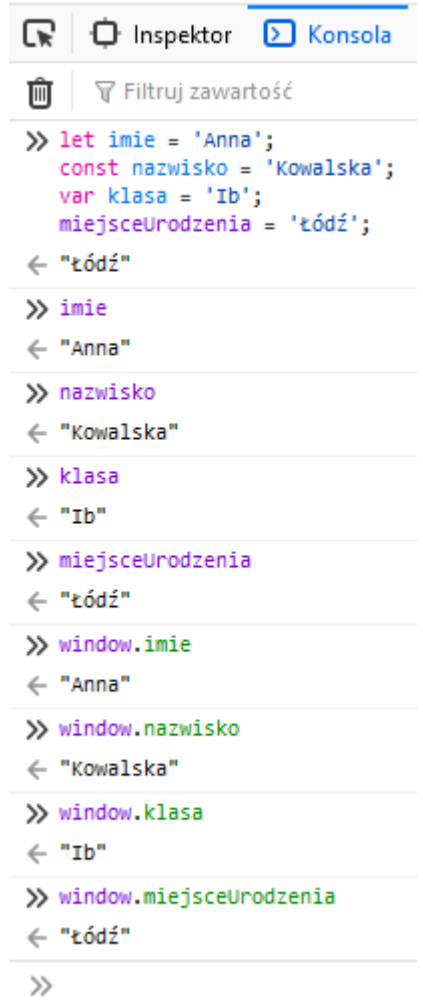
zmienne globalne

zmiennych globalnych
nie powinno się używać!
(mogą m.in. nadpisać pola
obiektu Window)

zmienne globalne zadeklarowane poza funkcją, mają zasięg widoczności całego skryptu, w którym zostały zadeklarowane

```
let imie = ’Anna’;
const nazwisko = „Kowalska”;
var klasa = „Ib”;
miejsceUrodzenia = „Łódź”;
```

do zmiennych globalnych można się dostać poprzez obiekt window



zmienne globalne zadeklarowane w funkcji

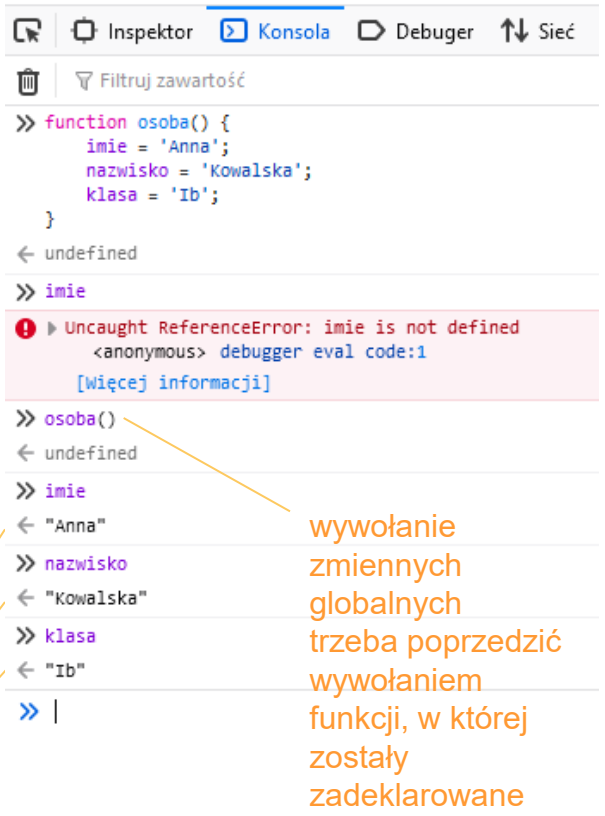
```
// zmienne imie, nazwisko, klasa są tutaj widoczne (poza funkcją)

function osoba() {

    imie = „Anna";
    nazwisko = „Kowalska”;
    klasa = „Ib”;

    // zmienne imie, nazwisko, klasa są tutaj widoczne
    // (pomiędzy nawiasami klamrowymi)

}

// zmienne imie, nazwisko, klasa są tutaj widoczne (poza funkcją)
```

zmienne globalne
imie, nazwisko ,klasa
są widoczne w całym skrypcie, w którym zostały zadeklarowane

zmiennych globalnych nie powinno się używać! (mogą m.in. nadpisać pola obiektu Window)

zmienne bez słów kluczowych

zasięg zmiennych można przetestować w konsoli (F12)

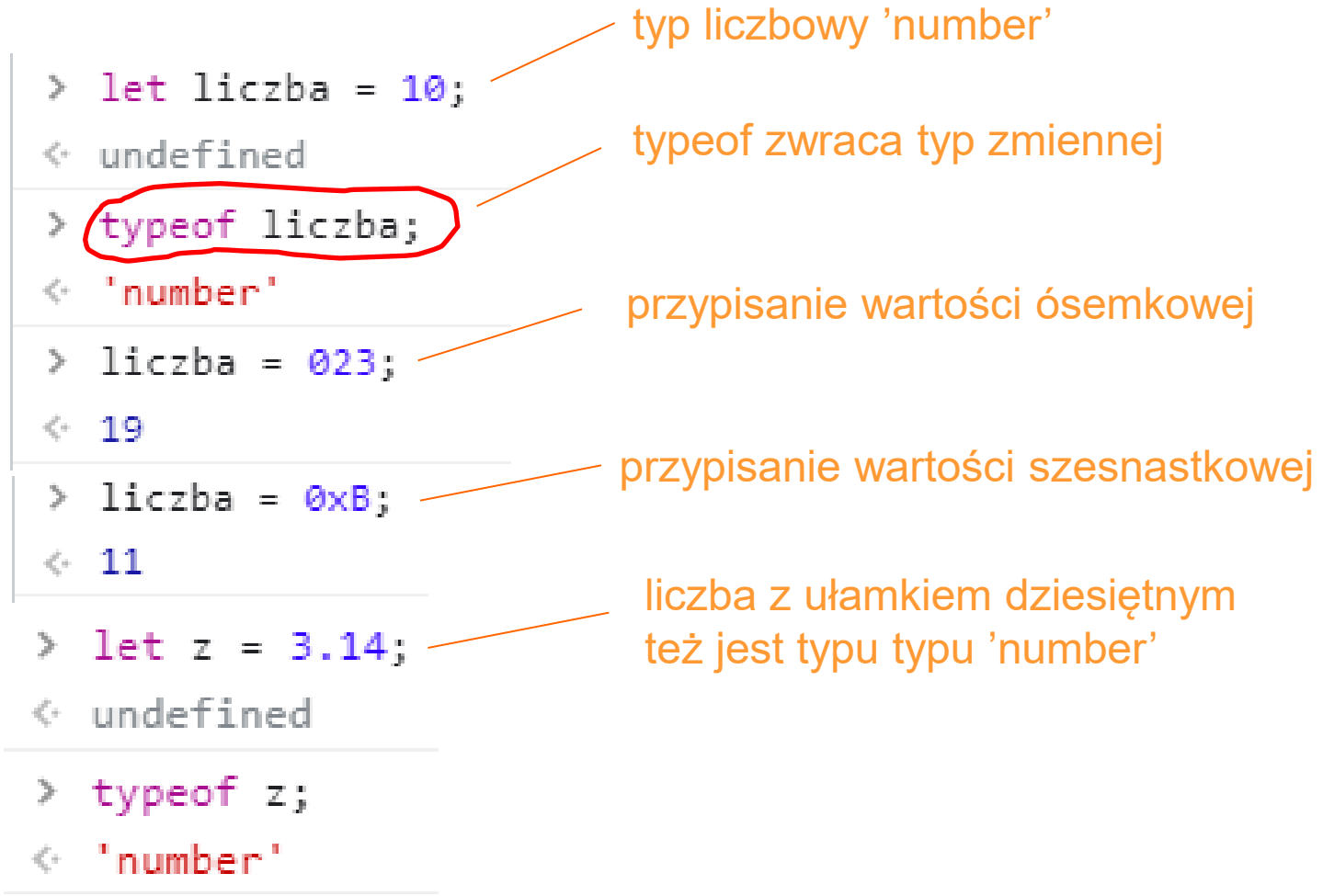


wywołanie zmiennych globalnych trzeba poprzedzić wywołaniem funkcji, w której zostały zadeklarowane

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

typ liczbowy ’number’

konsola (narzędzia developerskie F12)



typ łańcuchowy ’string’

typ ’string’ przechowuje łańcuchy znaków

```
> const napis = "Hello";
< undefined
> typeof napis;
< 'string'
```

typ łańcuchowy ’string’

łańcuch tekstowy (string)

```
tekst_1="Informatyka ";

tekst_2 ="jest OK";

tekst=tekst_1 + tekst_2;

document.write(tekst);
```

Informatyka jest OK

stringi można dodawać
(konkatenacja stringów)

Dodawanie stringów i liczb

```
x=5+5;
document.write(x);
```

10

```
x="5"+"5";
document.write(x);
```

55

```
x=5+"5";
document.write(x);
```

55

```
x="5"+5;
document.write(x);
```

55

Jeśli dodajemy łańcuch tekstowy i liczbę rezultatem będzie łańcuch

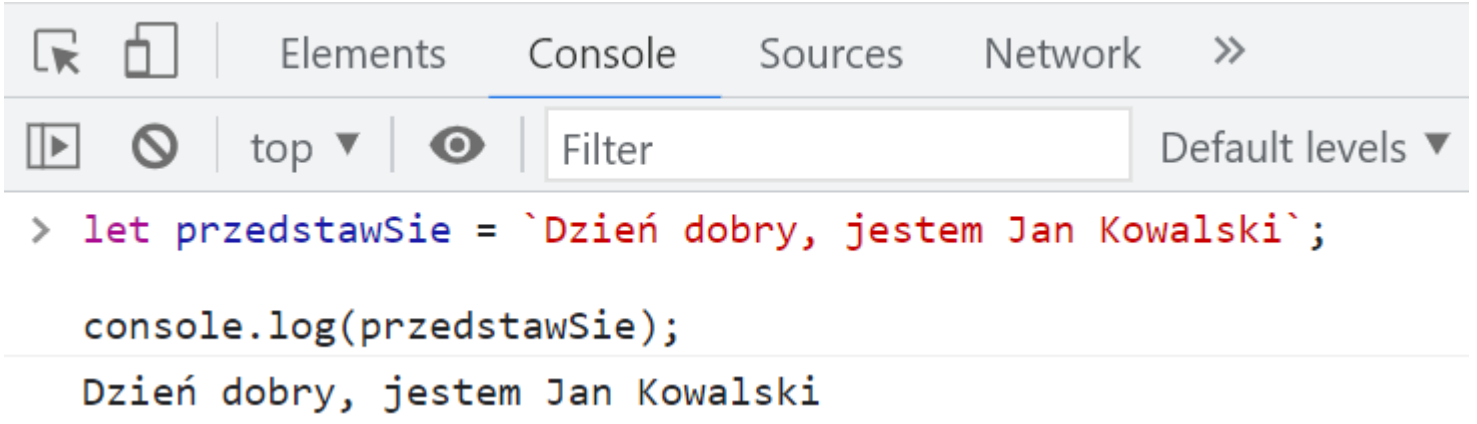
Try it Yourself

Template Literals (String Templates)

string template używa znaków backtick
(odwrócony apostrof, na klawiaturze pod klawiszem Escape)

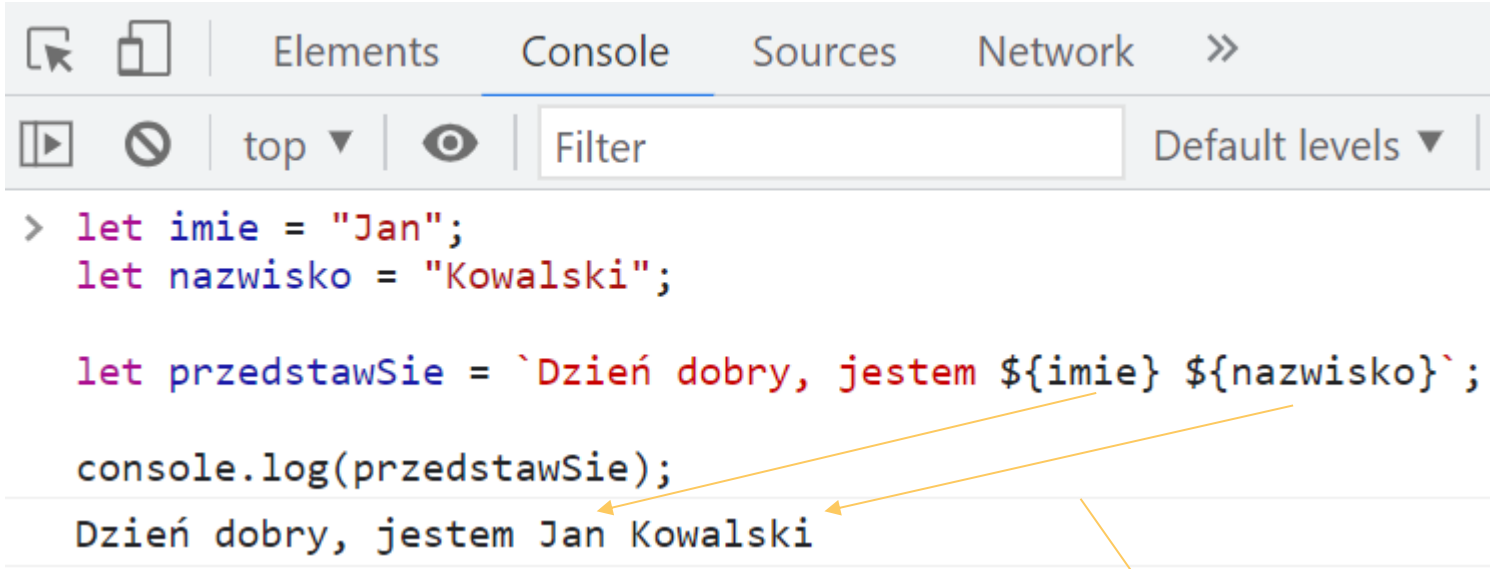
```
let przedstawSie = `Dzień dobry, jestem Jan Kowalski`;
```

string template



String Templates

Template Literals (String Templates)



String Templates

typ logiczny ’boolean’

typ logiczny może przyjmować wartości true lub false

```
> let prawda = true;
< undefined
> prawda
< true
> if(prawda) console.log("hello");
  hello
```

typy: null, undefined

wartość pusta jest typu ’object’

```
> const a = null;
<· undefined
> typeof a;
<· 'object'
```

wartość niezdefiniowana

```
> var c; typeof c;
<· 'undefined'
```

typ ’undefined’

złożone typy danych
obiekty

obiekt uczen jest odzwierciedleniem rzeczywistego ucznia - Jana Kowalskiego

obiekt napisany przez programistę

```
const uczen = {
   imie: "Jan",
   nazwisko: "Kowalski",
   klasa: "1TP",
   kolorOczu: "niebieski",
   wiek: 15
};
```

właściwości obiektu uczen

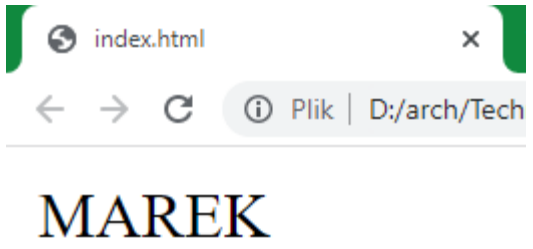
złożone typy danych
obiekty wbudowane

obiekt wbudowany
w przeglądarkę

String – manipulowanie ciągami tekstowymi

metoda obiektu String

```
let imie = "marek";
let imieDuzymiLiterami = imie.toUpperCase();
document.write(imieDuzymiLiterami);
```



złożone typy danych
obiekty wbudowane

obiekty wbudowane
w przeglądarkę

Array - służy do przechowywania wielu wartości w jednej zmiennej.

Date – służy do manipulowania datami i czasem

Math – umożliwia wykonywanie zadań matematycznych.

złożone typy danych
tablice

tablice służą do przechowywania wielu wartości

```
const imiona = ["Anna", "Bożena", "Tomasz"];

const imiona = new Array("Anna", "Bożena", "Tomasz");
```

2 sposoby utworzenia tablicy *imiona* przechowującej 3 imiona

złożone typy danych
tablica jednowymiarowa

deklaracja tablicy

```
let zeszyty = new Array();

zeszyty[0] = "matematyka";
zeszyty[1] = "biologia";
zeszyty[2] = "informatyka";
```

inizjalizacja tablicy

```
let zeszyty = new Array("matematyka ", "biologia ", "informatyka");
```

deklaracja z inicjalizacją

Operatory arytmetyczne

+	dodawanie	*addition*
-	odejmowanie	*subtraction*
*	mnożenie	*multiplication*
/	dzielenie	*division*
%	modulo – reszta z dzielenie liczb całkowitych	*modulo*

Operatory inkrementacji oraz dekrementacji

przykłady dla x=4;

++x	pre-inkrementacja	y=++x;	y=5; x=5;
--x	pre-dekrementacja	y=--x;	y=3; x=3;
x++	post-inkrementacja	y=x++;	y=4; x=5;
x--	post-dekrementacja	y=x--;	y=4; x=3;

```
<script type=”text/javascript”>
   x=4;
   y=x--;
   document.write(y + "  " + x);
</script>
```

skrypt sprawdzający operatory

Operatory przypisania złożonego

przykłady dla y=10

operator	przypisanie	inny zapis tego samego	rezultat
=	y=5		y=5
+=	y+=5	y=y+5	y=15
-=	y-=5	y=y-5	y=5
/=	y/=5	y=y/5	y=2
*=	y*=5	y=y*5	y=50
%=	y%=5	y=y%5	y=0

Operatory relacji (porównania)

==	równy	*Equal to*
===	dokładnie równy (wartość i typ)	*Strict equal*
!=	różny	*Not equal to*
!==	różny (wartość lub typ)	*Strict not equal*
>	Większy	*Greater than*
<	Mniejszy	*Less than*
>=	większy lub równy	*Greater than or equal to*
<=	mniejszy lub równy	*Less than or equal to*

x=5

Operatory relacji (porównania)

==	x==4	*fałsz*
	x==5	*prawda*
	x=="5"	*prawda*
===	x===5	*prawda*
	x==="5"	*fałsz*
!=	x!=5	*fałsz*
!==	x !== "5"	*prawda*
>	x>4	*prawda*
<	x<4	*fałsz*
>=	x>=4	*prawda*
<=	x<=4	*fałsz*

Operatory bitowe *Bitwise operators*

&	iloczyn bitowy AND	*Bitwise AND*
|	suma bitowa OR	*Bitwise inclusive OR*
^	bitowa różnica symetryczna XOR	*Bitwise exclusive OR*
~	negacja bitowa NOT	*Unary complement (bit inversion)*
>>	operator przesunięcia bitowego w prawo (zachowuje znak liczby)	*Signed right shift (Sign Preserving)*
<<	operator przesunięcia bitowego w lewo	*Zero fill left shift*
>>>	operator przesunięcia bitowego w prawo z wypełnieniem zerami (nie zachowuje znaku liczby)	*Zero fill right shift*

JavaScript Bitwise Operations w3schools

Operator bitowy AND

6 & 4 = ?

6 -> 110
4 -> 100

system binarny

system dziesiętny

x	y	x & y
0	0	0
0	1	0
1	0	0
1	1	1

Tablica prawdy dla AND

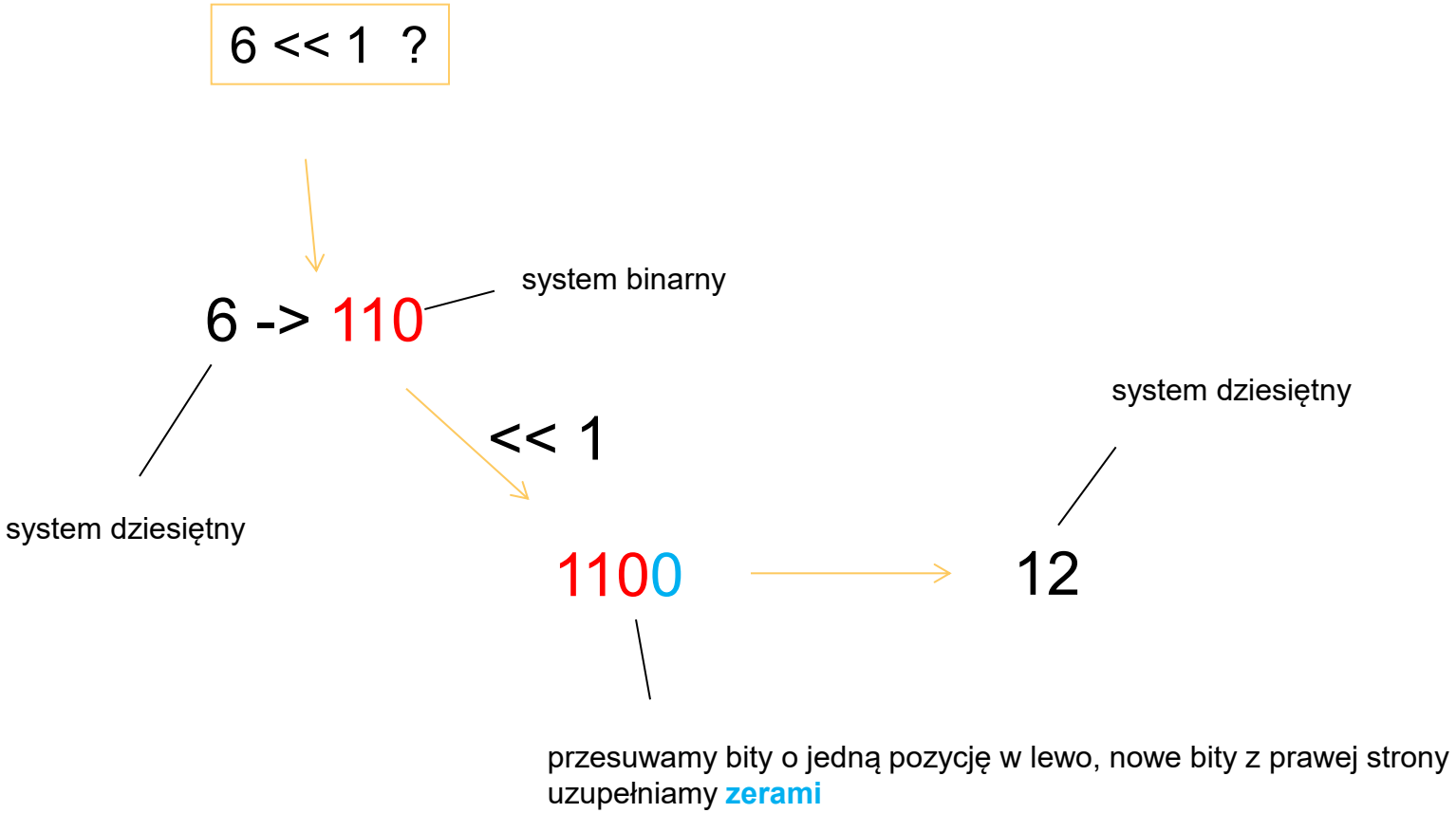
```
  110
&100
-------
  100
```

system dziesiętny

4

https://pl.wikipedia.org/wiki/Tablica_prawdy

Operator przesunięcia bitowego w lewo



Operatory bitowe w js

JavaScript przechowuje liczby jako 64-bitowe liczby zmiennoprzecinkowe, ale wszystkie operacje bitowe są wykonywane na 32-bitowych liczbach binarnych.

Przed wykonaniem operacji bitowej JavaScript konwertuje liczby na 32-bitowe liczby całkowite ze znakiem.

Po wykonaniu operacji bitowej wynik jest konwertowany z powrotem na 64-bitowe liczby JavaScript.

typ Number w js

reprezentacja binarna liczby zmiennoprzecinkowej - IEEE 754

system reprezentacji liczb całkowitych – kod uzupełnień do 2

U2 - liczby ujemne w systemie binarnym

Operator przesunięcia bitowego w prawo

bit znaku liczby:
0 – liczba dodatnia
1 – liczba ujemna

zachowuje bit znaku liczby niezmieniony

5

00000000000000000000000000000101

5 >> 1

00000000000000000000000000000010 (2)

taki sam wynik (dla liczb dodatnich)

5

00000000000000000000000000000101

5 >>> 1

00000000000000000000000000000010 (2)

nie następuje zmiana znaku liczby

-5

11111111111111111111111111111011

-5 >> 1

11111111111111111111111111111101 (-3)

różny wynik (dla liczb ujemnych)

-5

11111111111111111111111111111011

-5 >>>1

01111111111111111111111111111101
(2147483645)

następuje zmiana znaku liczby

Unsigned right shift (>>>)

Operatory logiczne

Logical operators

&&	koniunkcja (i)	*and*
ll	alternatywa (lub)	*or*
!	negacja (nie)	*not*

Operatory logiczne

Logical operators

x=6 i y=3

&&	(x < 8 && y > 4)	*fałasz*
II	(x==8 || y==3)	*prawda*
!	!(x==y)	*prawda*

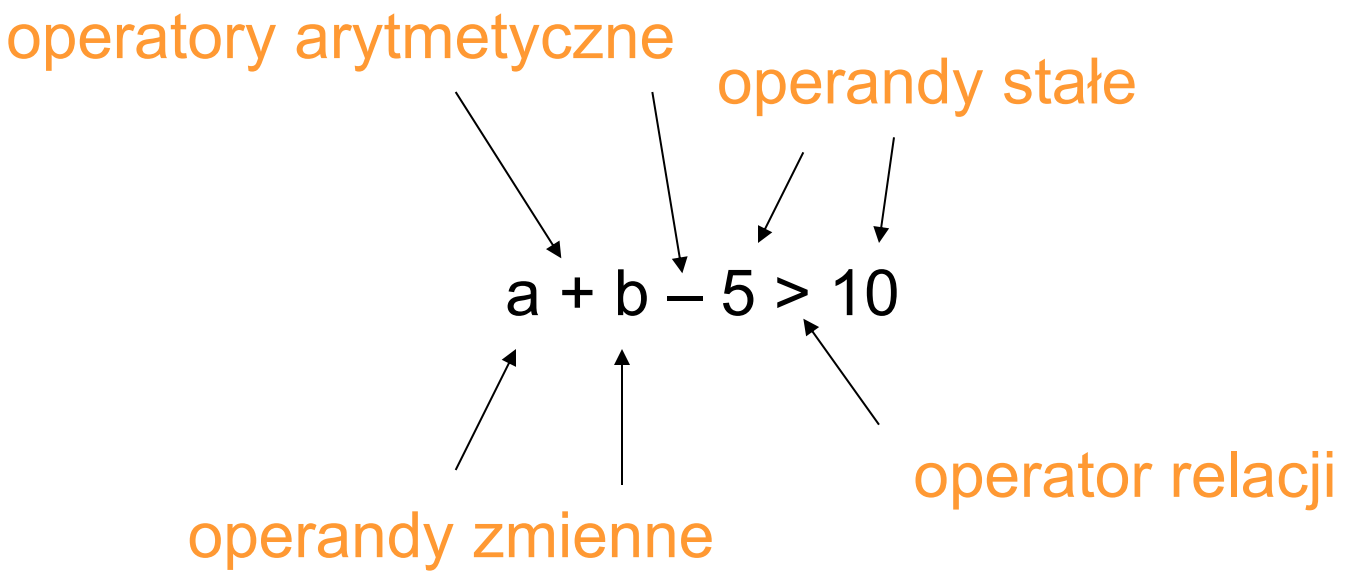
Znaki specjalne

escape sequences

znak	znaczenie	nazwa angielska	przykład
\n	nowy wiersz	*new-line NL (LF)*	
\t	tabulacja pozioma	*horizontal tab HT*	
\b	cofnięcie kursora o jeden znak	*backspace BS*	
\r	powrót karetki	*carriage return CR*	
\f	nowa strona	*form feed FF*	
\\	backslash (lewy ukośnik)	*backslash *	
\’	apostrof	*single quote ’*	
\"	cudzysłów	*double quote "*	

INSTRUKCJE STERUJĄCE

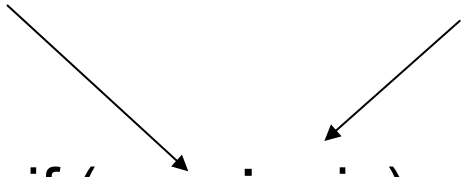
Wyrażenie


operatory arytmetyczne
operandy stałe
a + b – 5 > 10
operandy zmienne
operator relacji

Instrukcja warunkowa if

Wyrażenie może przyjąć jedną
z dwóch wartości logicznych:
true - wyrażenie jest prawdziwe
false – wyrażenie jest fałszywe

Podobnie jak w C++
przyjmuje się, że:
0 – false
liczba różna od zera – true

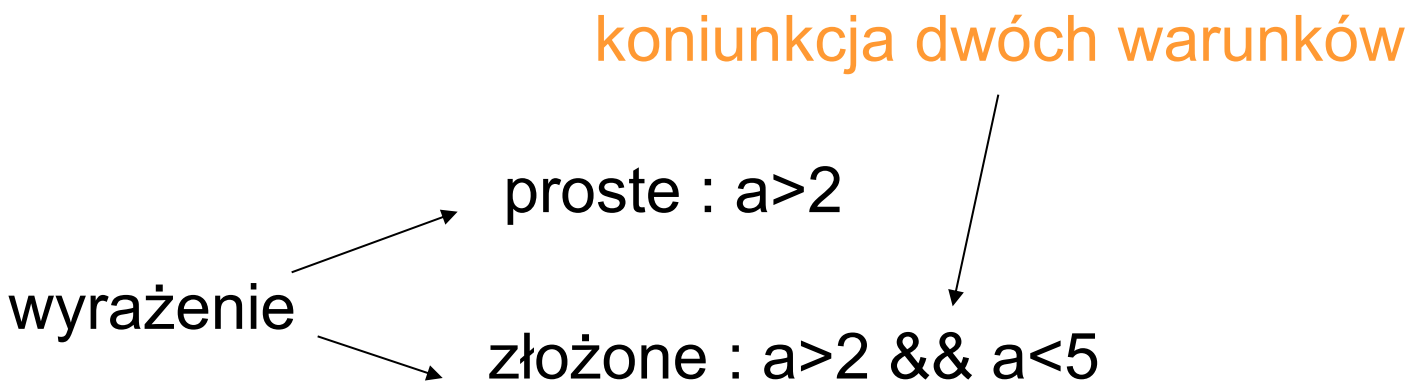


```
if (wyrażenie)
        instrukcja1
```

Przykłady:

(a==2)	true dla zmiennej a równej 2
	false dla zmiennej a równej 3
(a)	true dla zmiennej a różnej od 0 (np.: 1,2,4, 10 itp.)
	false dla zmiennej a równej 0
a+2*b>c	true jeśli wartość wyrażenia a+2*b jest wieksza od c

Instrukcja warunkowa if



Instrukcja warunkowa if

```
if (wyrażenie)
        instrukcja1
```

Pojedyncza instrukcja
nie musi mieć nawiasów
klamrowych

```
if (wyrażenie)
        instrukcja1
else
        instrukcja2
```

```
if (wyrażenie) {
        instrukcja1
        instrukcja2
        instrukcja3
        }
```

blok instrukcji
(musi być ujęty
w nawiasy klamrowe)

```
if (wyrażenie 1)
        instrukcja1
else if (wyrażenie 2)
        instrukcja2
else if (wyrażenie 3)
        instrukcja3
```

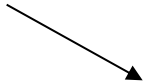
```
if (wyrażenie 1)
        instrukcja1
else if (wyrażenie 2)
        instrukcja2
else if (wyrażenie 3)
        instrukcja3
else
        instrukcja4
```

wybór wielowariantowy

Instrukcja warunkowa if

```
if (wyrażenie)
        instrukcja1
    else
        instrukcja2
```

inny zapis
Instrukcji
warunkowej



```
zmienna=(wyrażenie)? instrukcja1 : instrukcja2
```

Instrukcja switch

```
switch(wyrażenie)
{
        case wartość1:
                        instrukcja1;
                        break;
        case wartość2:
                        instrukcja2;
                        break;
        default:
                        instrukcja3;
                        break;
}
```

Pętla for

Przed pierwszym uruchomieniem pętli wykonywana jest ta instrukcja

Gdy jest tylko jedna instrukcja w treści pętli nawiasów może nie być

Warunek pętli – wyrażenie sprawdzane przed każdorazowym obiegiem pętli. Jeśli jest prawdziwe wykonywana jest treść pętli.

Instrukcja wykonywana każdorazowo na zakończenie obiegu pętli

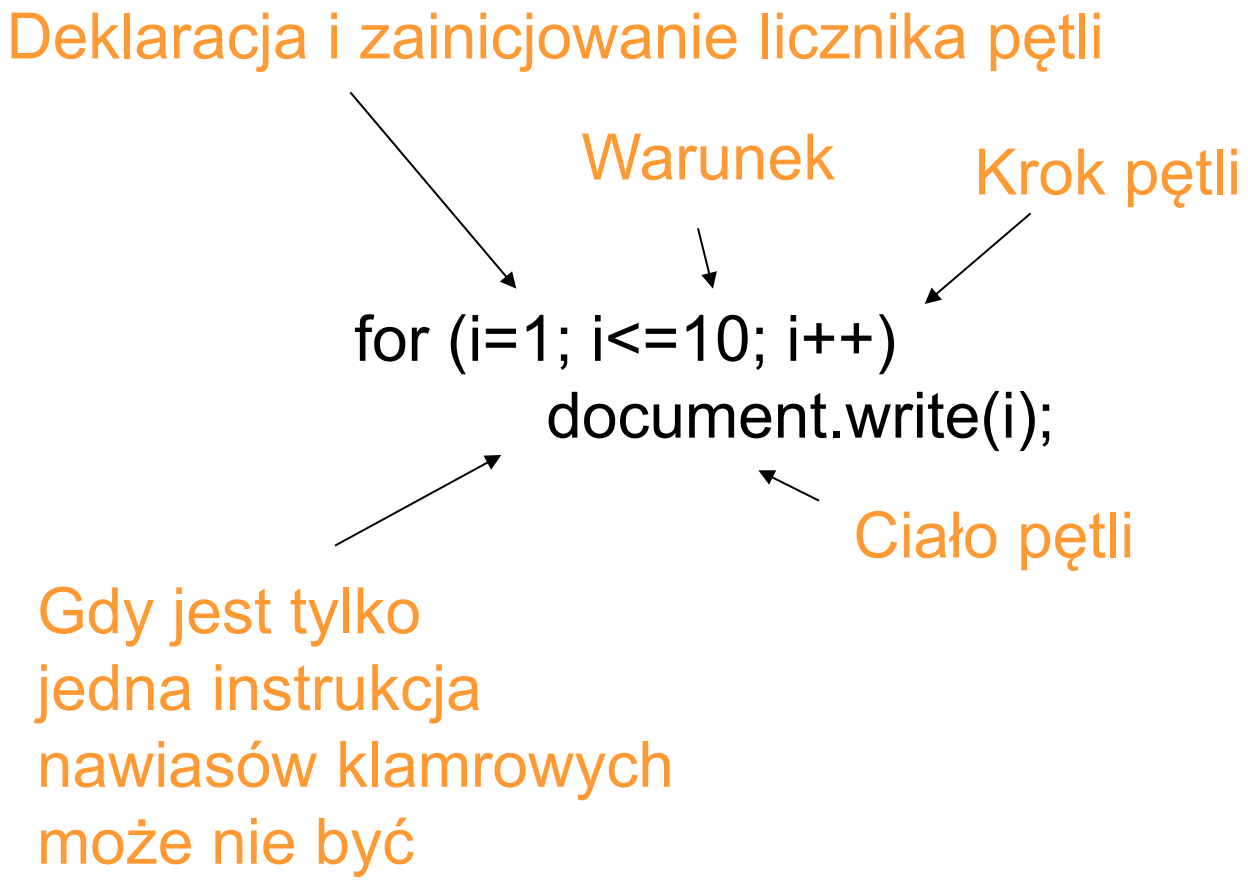
```
for (i=1; i<=10; i++)
{
   document.write(”Licznik pętli: ”);
   document.write (i);
   document.write(”<br>”);
}
```

Treść pętli (instrukcje ujęte w nawiasach klamrowych)

kod można zapisać w jednej linijce

```
for (i=1; i<=10; i++)
{
   document.write("Licznik pętli: " + i + "<br>");
}
```

Pętla for


Deklaracja i zainicjowanie licznika pętli
Warunek
Krok pętli
for (i=1; i<=10; i++)
document.write(i);
Ciało pętli
Gdy jest tylko
jedna instrukcja
nawiasów klamrowych
może nie być

Pętla for

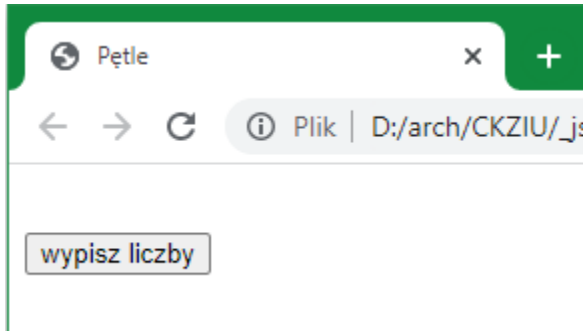
Kolejność wykonywania instrukcji

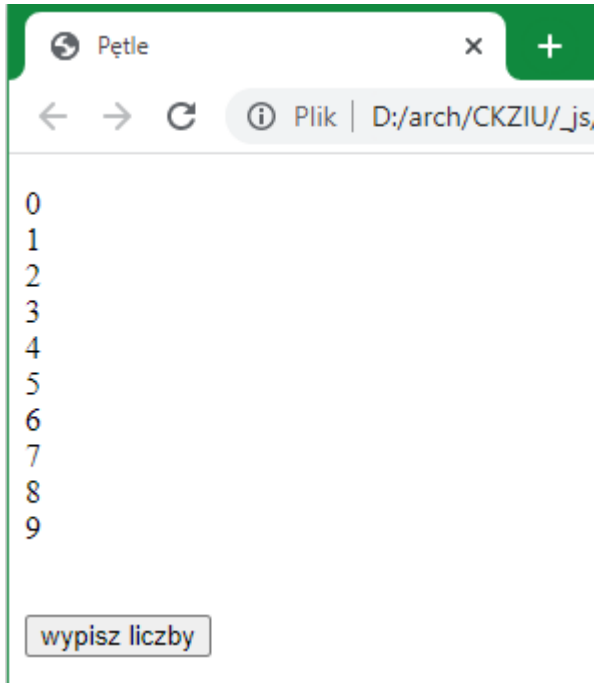
```
        1       2     4
for (i=1; i<=10; i++)
{
                3
    document.write(i);
}
```

pierwszy obieg pętli	1234
drugi obieg pętli	234
trzeci obieg pętli	234
kolejne obiegi pętli	234

Pętla for

wypisz liczby od 1 do 10
za pomocą pętli for



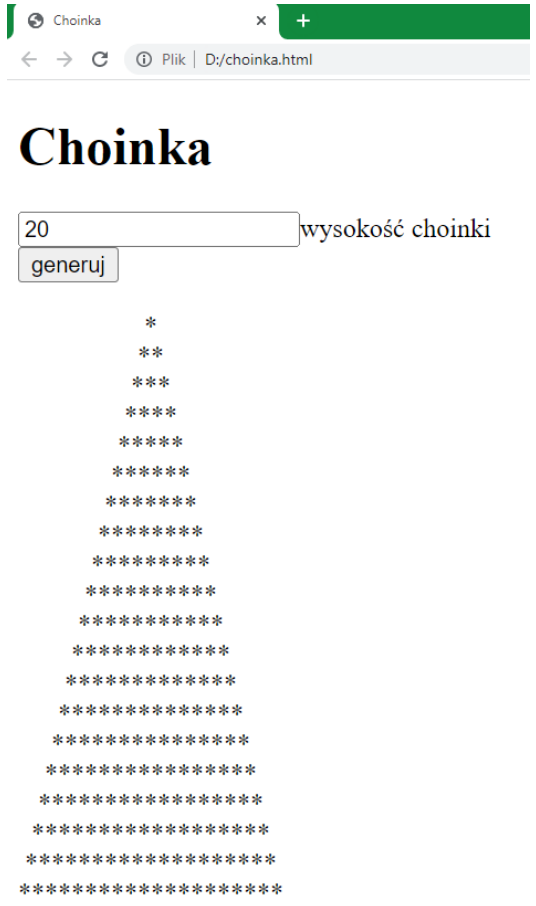


Ćwiczenia

1. Napisz skrypt wyświetlający na stronie tabelę o zadanej liczbie kolumn i rzędów. W polu tabeli umieść słowa „informatyka”.
2. Do poprzedniego zadania dodaj następujące funkcjonalności:
   - zadana treść tabeli
   - zadana grubość tabeli
   - zadany kolor tła

Ćwiczenia

3. Napisz skrypt generujący choinkę.



po naciśnięciu przycisku generowana jest choinka o zadanej wysokości

Choinka

choinka.html > ...

```
1   <!doctype html>
2   <html>
3   <head>
4       <meta charset="utf-8">
5       <title>Ankieta</title>
6       <script src='main.js'></script>
7   </head>
8   <body>
9   <h1>Choinka</h1>
10  <input id="wysokosc" type="text" >wysokość choinki<br>
11  <button id="btn" onclick="pokaz()">generuj</button>
12  <div id="obrazek"></div>
13
14  </body>
15  </html>
```

pole do wpisywania wysokości choinki

div z rysunkiem choinki

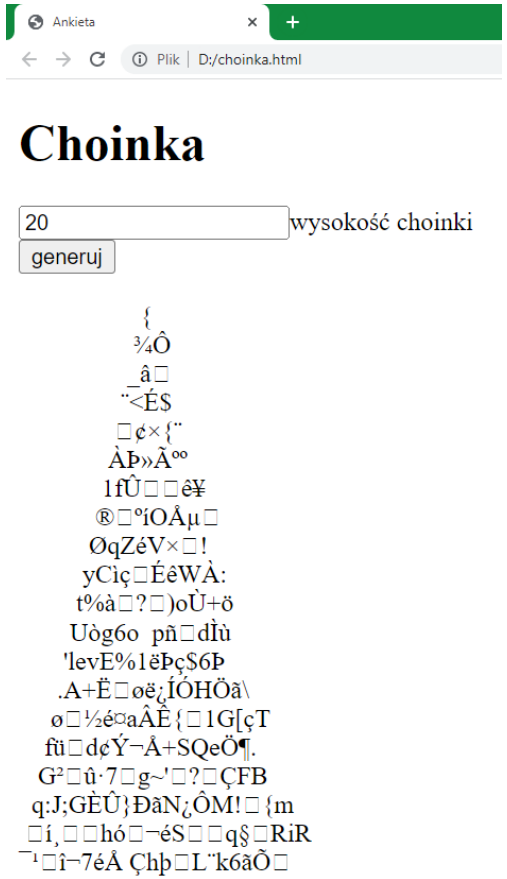
przycisk trigerujący funkcję pokaz()

Choinka


JS main.js > ...
1 function pokaz(){
2 let obrazek = document.getElementById("obrazek")
3 let h = document.getElementById("wysokosc").value;
4 h = parseInt(h);
5
6 let choinka = "";
7
8 for(let i = 0; i < h + 1 ; i++){
9 for(let k = i ; k < h ; k++){
10 choinka += " ";
11 }
12 for(let j = 0; j < i ; j++){
13 choinka += "*";
14 }
15 choinka += "
"
16 }
17 obrazek.innerHTML = choinka;
18 }
zamiana stringa na int
zmienna, do której wpisujemy elementy choinki (tworzymy obrazek choinki)
pętla przełączająca poszczególne piętra choinki
pętla rysująca spacje na danym piętrze
pętla rysująca choinkę na danym piętrze
gdy rysowanie danego pietra zostanie zakończone przechodzimy do nowej linii
wrzucamy wyrysowaną choinkę na stronę

Choinka

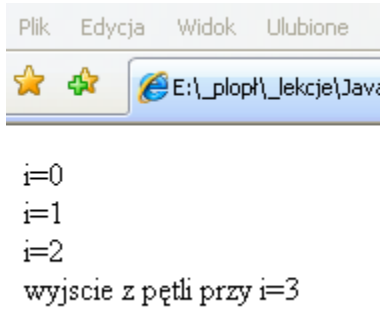
3. Napisz skrypt generujący choinkę z losowych znaków.



break

```
<script>
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
    break;
    }
  document.write("i=" + i);
  document.write("<br />");
  }
document.write(" wyjście z pętli przy i=" + i);
</script>
```

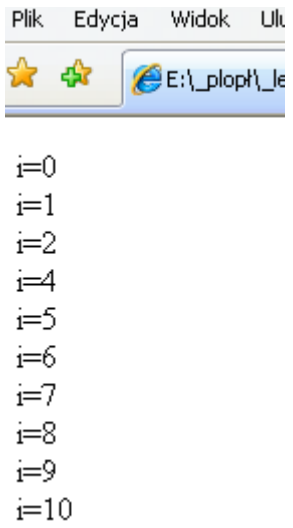
break przerywa pętlę
(wyjście z pętli for przy i = 3)



continue

```
<script>
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
    continue;
    }
  document.write(„i=" + i);
  document.write("<br />");
  }
</script>
```

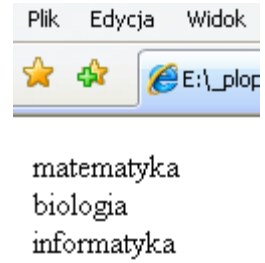
przy i = 3 przerywa bieżącą pętlę
(continue pomija bieżącą kolejkę)



brak i=3

Pętla for …in

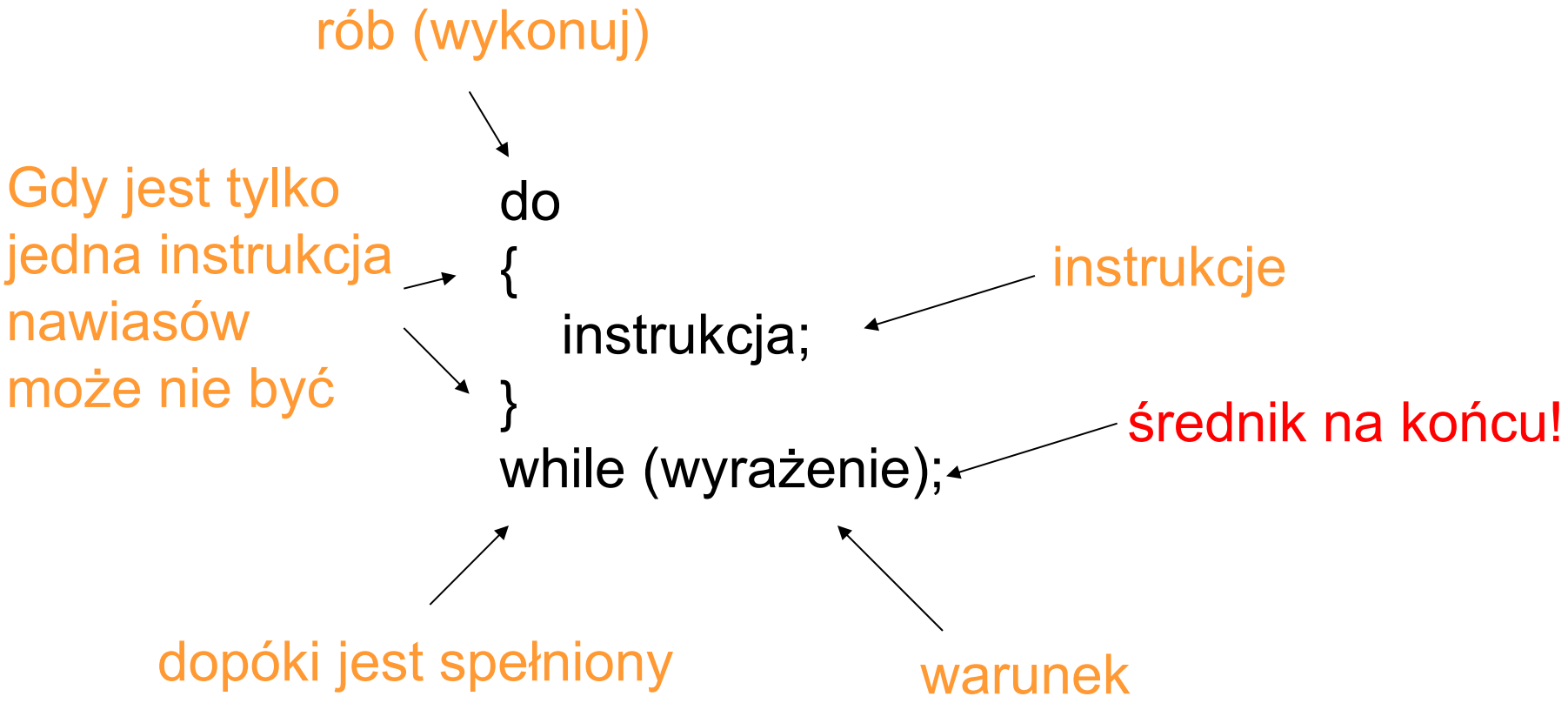
```
<html>
<body>
<script>
var x;
var zeszyty = new Array();
zeszyty[0] = "matematyka";
zeszyty[1] = "biologia";
zeszyty[2] = "informatyka";

for (x in zeszyty)
{
document.write(zeszyty[x] + "<br />");
}
</script>
</body>
</html>
```



Pętla while


dopóki jest spełniony
warunek
gdy jest tylko
jedna instrukcja
nawiasów
może nie być
while (wyrażenie)
{
instrukcja;
}
wyrażenie może przyjmować
wartość logiczną:
• prawda
• fałsz
wykonuj instrukcje
(może ich być wiele)
brak średnika na końcu!

Pętla do while



wyrażenie może przyjmować wartość logiczną:

- prawda
- fałsz

Łamanie linii

```
document.write("Ten tekst napisała funkcja wypisz()");
```

Gdy chcesz złamać linie (w kodzie) wstaw w stringu \ (backslash)

```
document.write("Ten tekst napisała \
funkcja wypisz()");
```

Funkcje

```
<html>
<head>
<script>

function wypisz()
 {
     document.write("Ten tekst napisała funkcja wypisz()");
 }

</script>
</head>
<body>

<input type="button" onclick="wypisz()" value="zastosuj funkcję wypisz()" />

</body>
</html>
```

funkcja
(deklaracja i definicja)

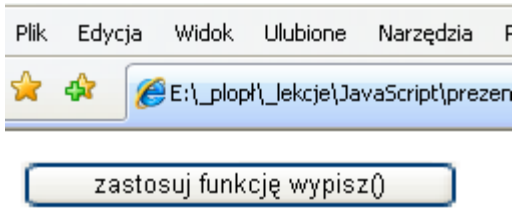
ciało funkcji

zdarzenie

wywołanie funkcji

przycisk

napis na przycisku



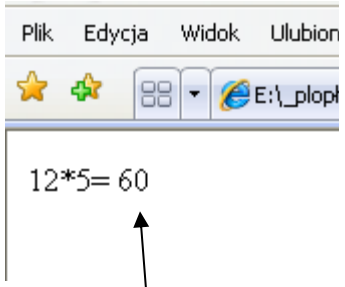
Ten tekst napisała funkcja wypisz()

Funkcja przyjmująca argumenty i zwracająca wartość

```
<html>
<head>
    <script>
        function mnozenie(a,b)
        {
           return a*b;
        }
    </script>
</head>

<body>
    12*5=

    <script>
       document.write(mnozenie(12,5));
    </script>

</body>
</html>
```

argumenty formalne

wartość zwracana przez funkcję
(gdy nie ma słowa kluczowego
return funkcja zwraca undefined)



wygenerowane
przez funkcję

argumenty aktualne

wywołanie funkcji mnozenie()

Funkcja przyjmująca dowolną ilość argumentów

```html
<html>
<head>
    <script>
        function scal()
        {
            let napis = "";
            for(let i = 0; i < arguments.length; i++){
                console.log( i + " argument: " + arguments[i]);
                napis += arguments[i];
            }
            document.write(napis);
        }
    </script>
</head>
<body>
    <input type="button" onclick="scal('ala', 'ma', 'kota')"
    value="scal wyrazy podane jako argumenty" />
</body>
</html>
```

tablica argumentów funkcji

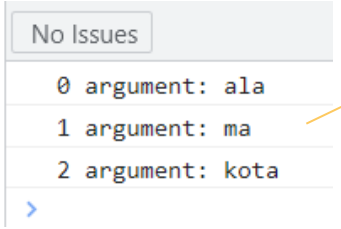
zmienna lokalna (widoczna tylko w funkcji)

scalenie argumentów w jeden wyraz

arguments[0] arguments[1] arguments[2]

scal wyrazy podane jako argumenty

alamakota



argumenty funkcji w logach konsoli

Funkcja przyjmująca dowolną ilość argumentów

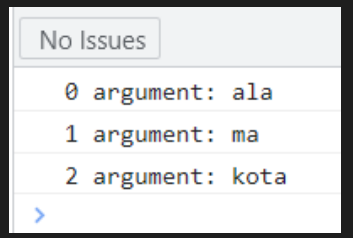
```
<html>
<head>
    <script>
        function scal()
        {
            napis = "";
            for(let i = 0; i < arguments.length; i++){
                console.log( i + " argument: " + arguments[i]);
                napis += arguments[i];
            }
            return napis
        }
        scal('ala', 'ma', 'kota');
        document.write(napis);
    </script>
</head>
<body>
</body>
</html>
```

zmienna globalna
(zadeklarowana bez var, let, const)

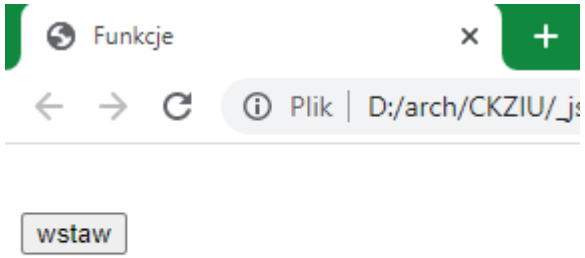
wywołanie funkcji

zmienna globalna jest widoczna poza funkcją
(musi być poprzedzona wywołaniem funkcji)

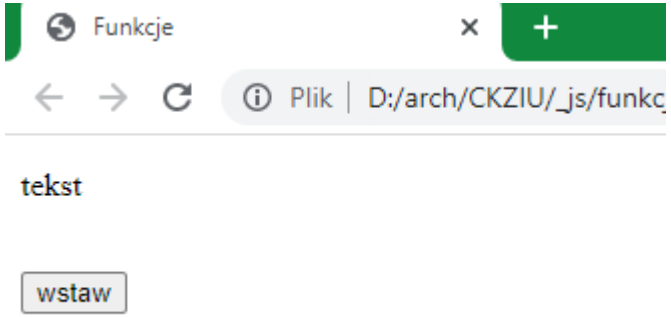
alamakota



Funkcje



stwórz przycisk, który wstawia tekst na stronę



Funkcje

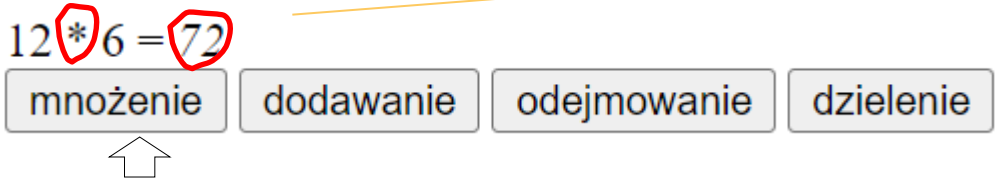
index.html

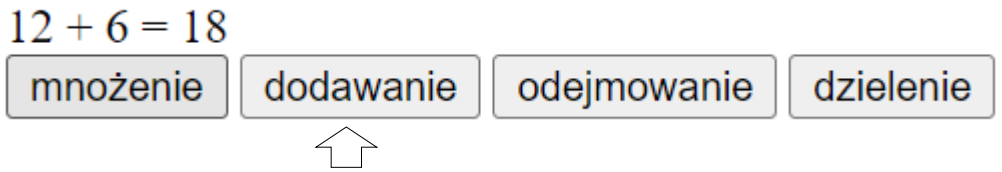
```html
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>Funkcje</title>
    <script src='main.js'></script>
</head>
<body>
    <p id="p1"></p><br>
    <button onclick="wstaw()">wstaw</button>
</body>
</html>
```

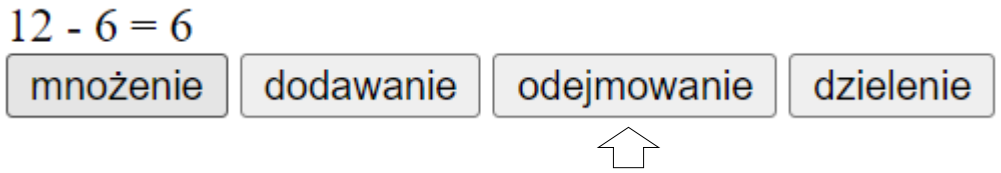
main.js

```js
function wstaw(){
    let p1 = document.getElementById("p1");
    p1.innerText="tekst";
}
```

Kalkulator przyciskowy







każdy z przycisków wstawia odpowiedni operator i wynik do liczb 12 oraz 6

Kalkulator przyciskowy

index.html

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>wstawianie</title>
    <!-- atrybut "defer" powoduje, że skrypt się wykona po załadowanoiu strony-->
    <script src='main.js' defer></script>
</head>
<body>
    12 <span id="operator"></span> 6 = <span id="wynik"></span><br>
    <button onclick="mnozenie(12,6)">mnożenie</button>
    <button onclick="dodawanie(12,6)">dodawanie</button>
    <button onclick="odejmowanie(12,6)">odejmowanie</button>
    <button onclick="dzielenie(12,6)">dzielenie</button>
</body>
</html>
```

każdy z przycisków wywołuje odpowiednią funkcję

Kalkulator przyciskowy

main.js

```
// żeby zmienne się utworzyły cała strona musi być załadowana
// trzeba dodać atrybut "defer" w <script> w pliku html
let wynik = document.getElementById("wynik");
let operator = document.getElementById("operator");

function mnozenie(a,b){
    //wynik.innerText=a*b;
    // alternatywnie można wstawić kod HTML
    wynik.innerHTML=`<span style="font-style: italic;">${a*b}</span>`;
    operator.innerText="*";
}

function dodawanie(a,b){
    wynik.innerText=a+b;
    operator.innerText="+";
}

function odejmowanie(a,b){
    wynik.innerText=a-b;
    operator.innerText="-";
}

function dzielenie(a,b){
    // zaokrąglenie do 2 liczb po przecinku
    wynik.innerText=(a/b).toFixed(2);
    operator.innerText="/";
}
```

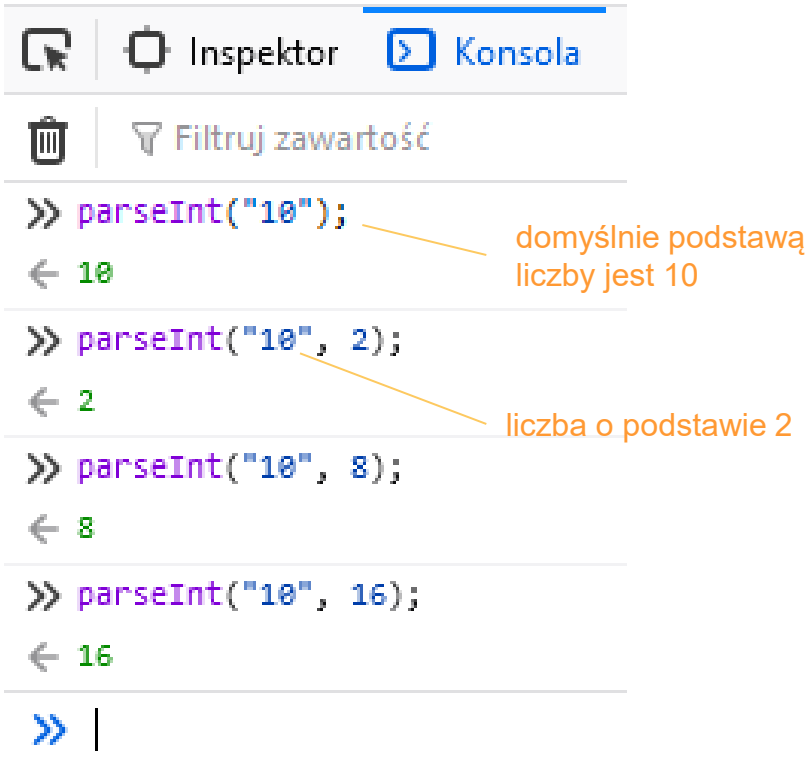
to nie zadziała bez atrybutu defer w <script>

definicje funkcji wywoływanych przez przyciski

funkcje wbudowane

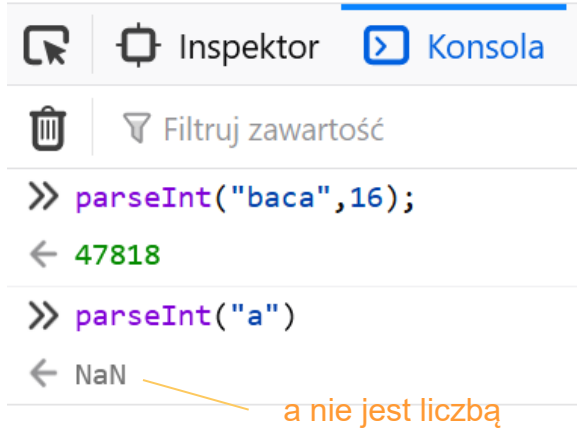
funkcja zamienia ciąg tekstowy będący liczbą o danej podstawie na liczbę całkowitą

```
parseInt("liczba", podstawa_liczby)
```

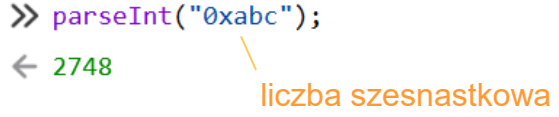


domyślnie podstawą liczby jest 10

liczba o podstawie 2



a nie jest liczbą



liczba szesnastkowa

funkcje wbudowane

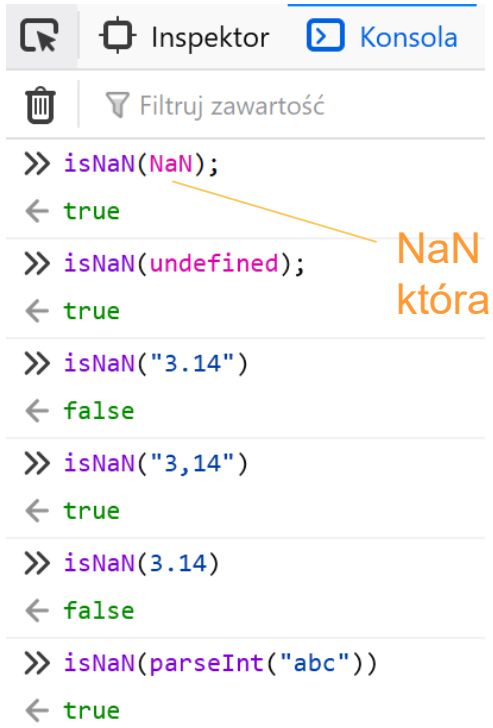
funkcja zamienia ciąg tekstowy będący liczbą (pierwszą napotkaną liczbę) na liczbę zmiennoprzecinkową

```
parseFloat("liczba")
```

funkcje wbudowane

funkcja sprawdza czy value nie jest liczbą

```
isNaN(value)
```

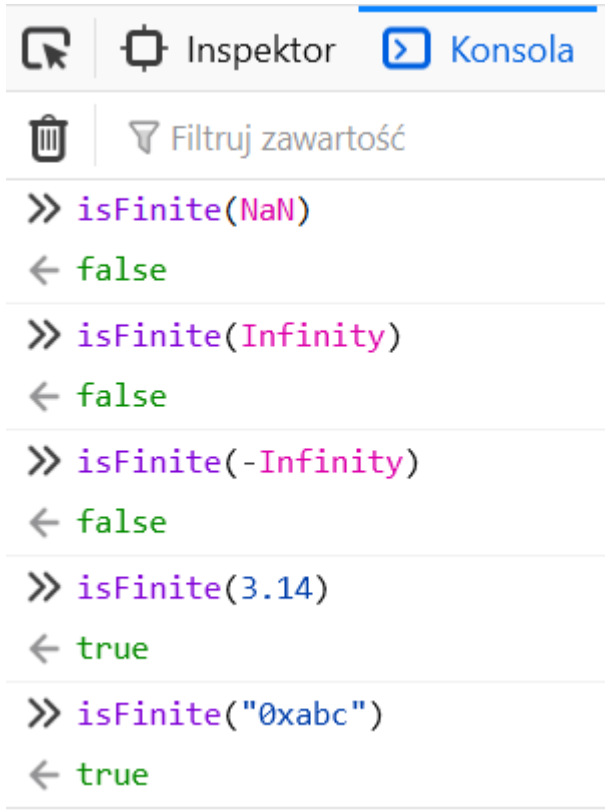


NaN oznacza Not-A-Number czyli wartość, która nie jest liczbą

funkcje wbudowane

funkcja sprawdza czy value jest różne od Infinity oraz od NaN

```
isFinite(value)
```



funkcje strzałkowe (arrow functions)

arrowFunctions > index.html > ...

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <meta charset='utf-8'>
5       <title>Arrow Functions</title>
6   </head>
7   <body>
8       <script>
9           const fun1 = function(something){
10              return something.toUpperCase();
11          }
12          const fun2 = (something) => {
13              return something.toUpperCase();
14          }
15          const fun3 = something => {
16              return something.toUpperCase();
17          }
18          const fun4 = (something) => something.toUpperCase();
19          const fun5 = something => something.toUpperCase();
20
21          document.write(fun1("fun1 "));
22          document.write(fun2("fun2 "));
23          document.write(fun3("fun3 "));
24          document.write(fun4("fun4 "));
25          document.write(fun5("fun5 "));
26
27      </script>
28  </body>
29  </html>
```

Funkcje strzałkowe pozwalają pisać krótszą składnię funkcji

tradycyjny zapis funkcji

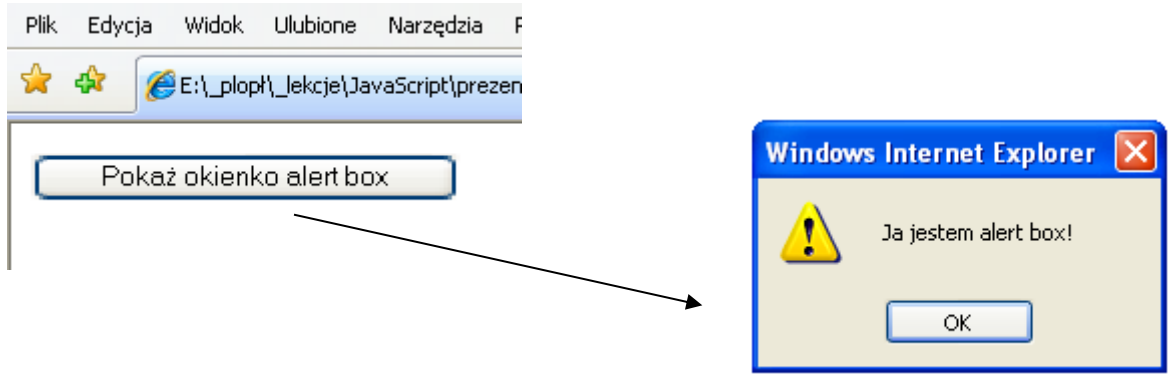
różna składnia równoważnych funkcji strzałkowych

wszystkie funkcje działają tak samo

FUN1 FUN2 FUN3 FUN4 FUN5

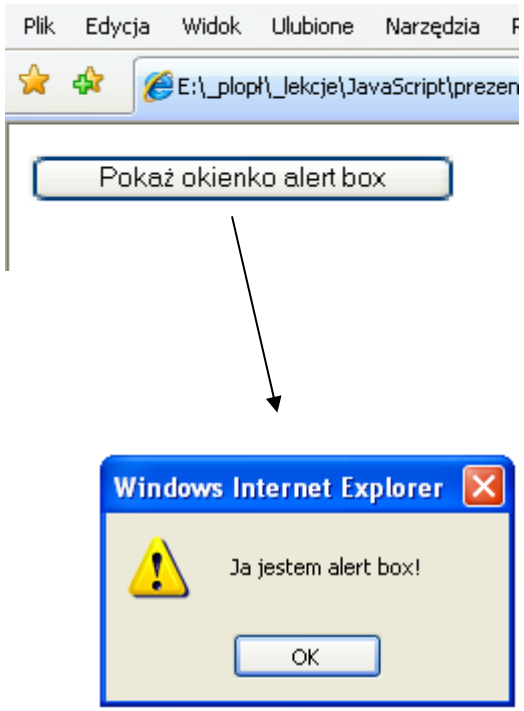
Alert box

```
<html>
<head>
</head>
<body>

<input type="button" onclick="alert('Ja jestem alert box!'); "
value="Pokaż okienko alert box" >

</body>
</html>
```



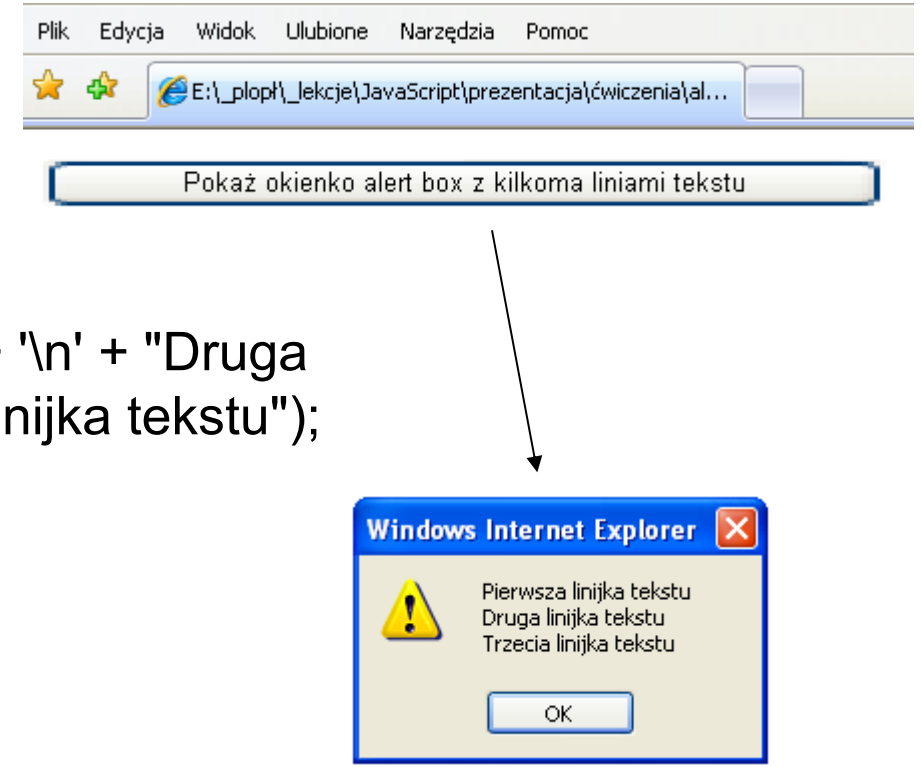
Alert box

```
<html>
<head>

    <script>
        function show_alert()
        {
           alert("Ja jestem alert box!");
         }
    </script>
</head>
<body>

    <input type="button"
       onclick="show_alert()"
       value="Pokaż okienko alert box" >

</body>
</html>
```



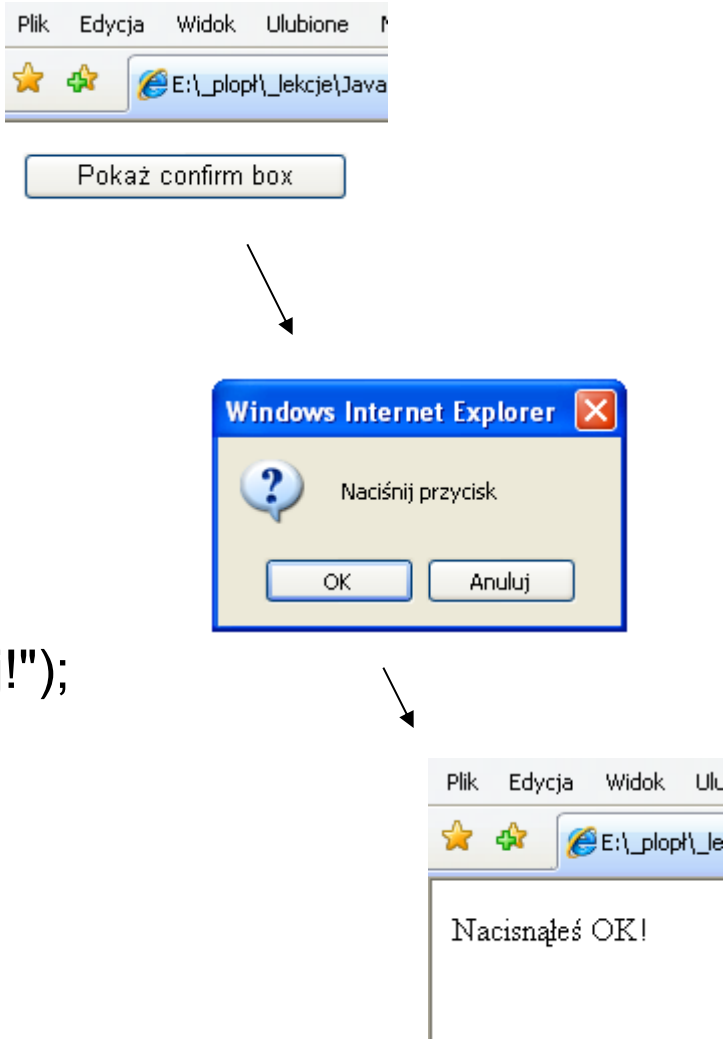
Alert box

```
<html>
<head>
<script>
    function pokaz_alert()
    {
        alert("Pierwsza linijka tekstu" + '\n' + "Druga
        linijka tekstu" + '\n' + "Trzecia linijka tekstu");
    }
</script>
</head>
<body>

    <input type="button" onclick="pokaz_alert()"
        value="Pokaż okienko alert box z kilkoma liniami
        tekstu" />

</body>
</html>
```



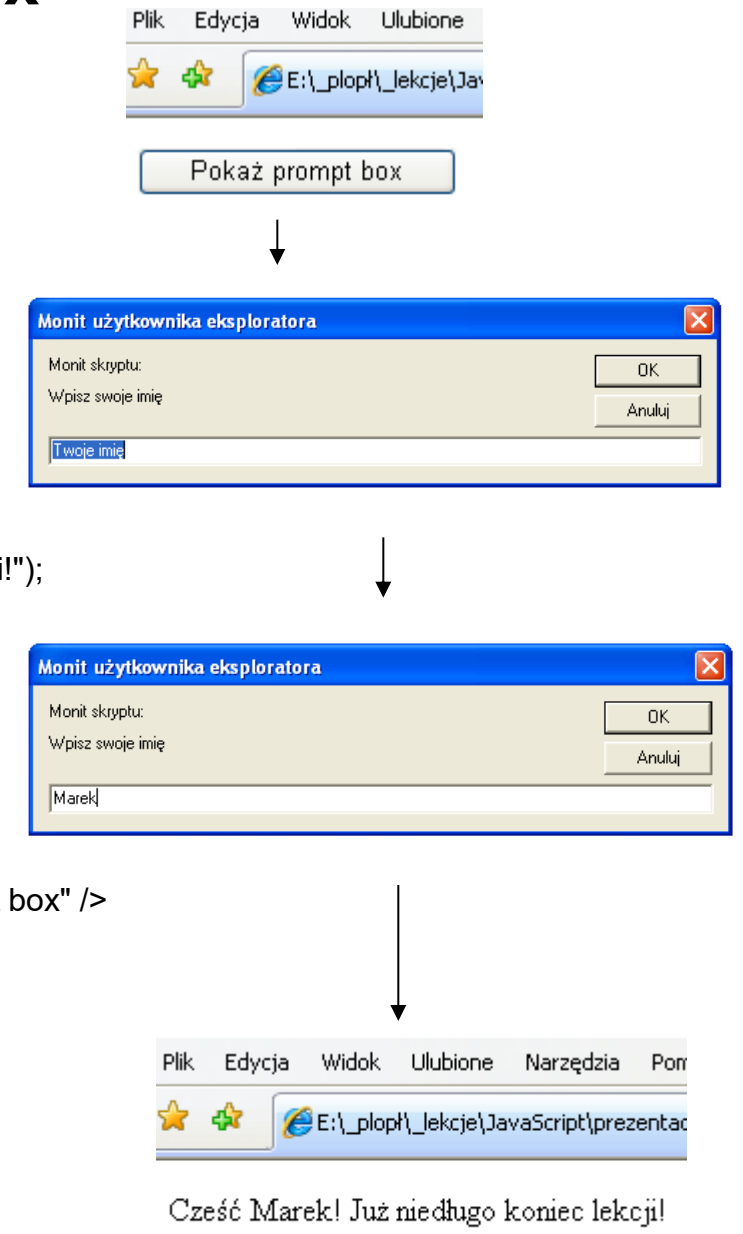
Confirm box

```
<html>
<head>
<script>
      function show_confirm()
      {
         let odp=confirm("Naciśnij przycisk");
         if (odp==true)
         {
           document.write("Nacisnąłeś OK!");
         }
         else
         {
           document.write("Nacisnąłeś Anuluj!");
         }
      }
</script>
</head>
<body>

         <input type="button" onclick="show_confirm()" value="Pokaż confirm box" />

</body>
</html>
```



Prompt box

```
<html>
<head>
<script>
        function show_prompt()
        {
            let imie=prompt("Wpisz swoje imię","Twoje imię");
            if (imie != null && imie!="")
            {
                document.write("Cześć " + imie + "! Już niedługo koniec lekcji!");
            }
        }
</script>
</head>
<body>

        <input type="button" onclick="show_prompt()" value="Pokaż prompt box" />

</body>
</html>
```



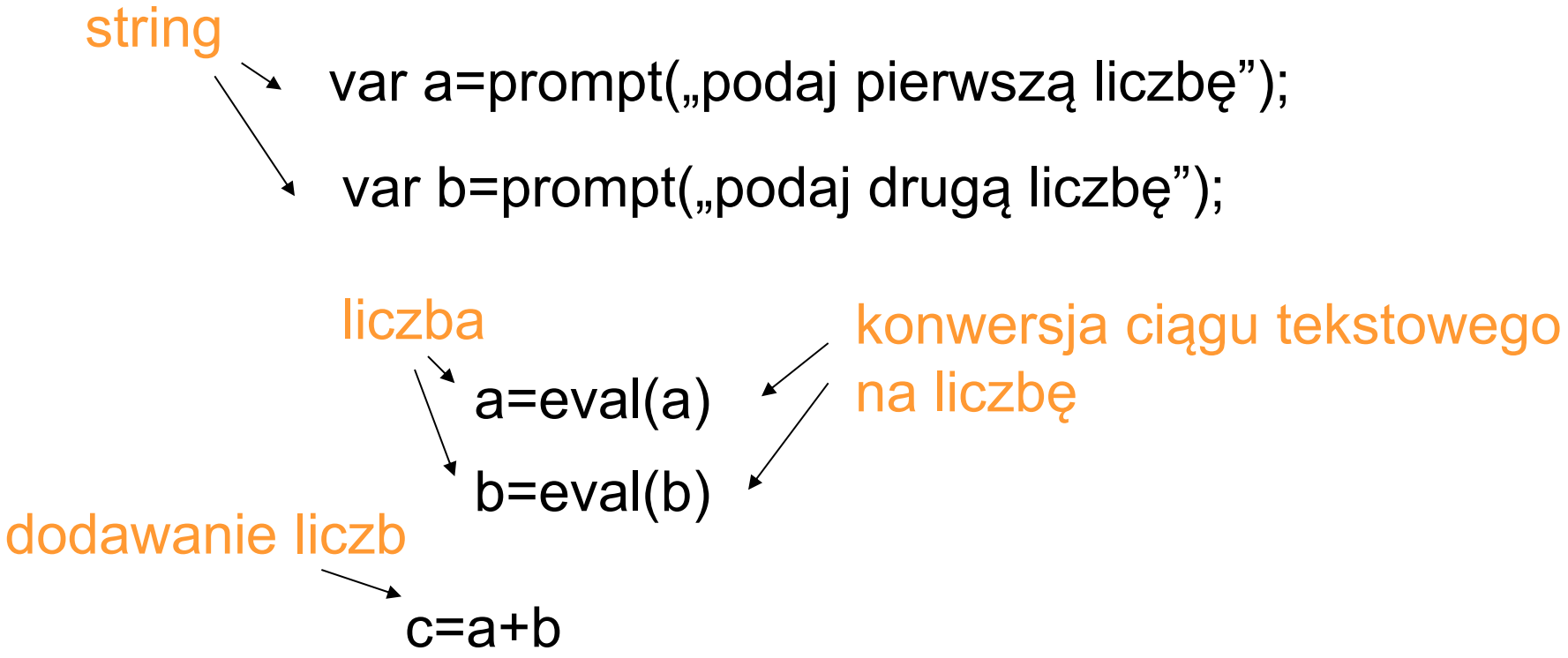
Ćwiczenia

1. Napisz sumator: podajemy dwie liczby
   i otrzymujemy ich sumę

2. Napisz pełny kalkulator z możliwością dodawania, odejmowania, mnożenia i dzielenia dwóch pobranych od użytkownika liczb.

Prompt box

Zamiana ciągu tekstowego na liczbę



Obiekty

obiekt **informatyk**

Properties

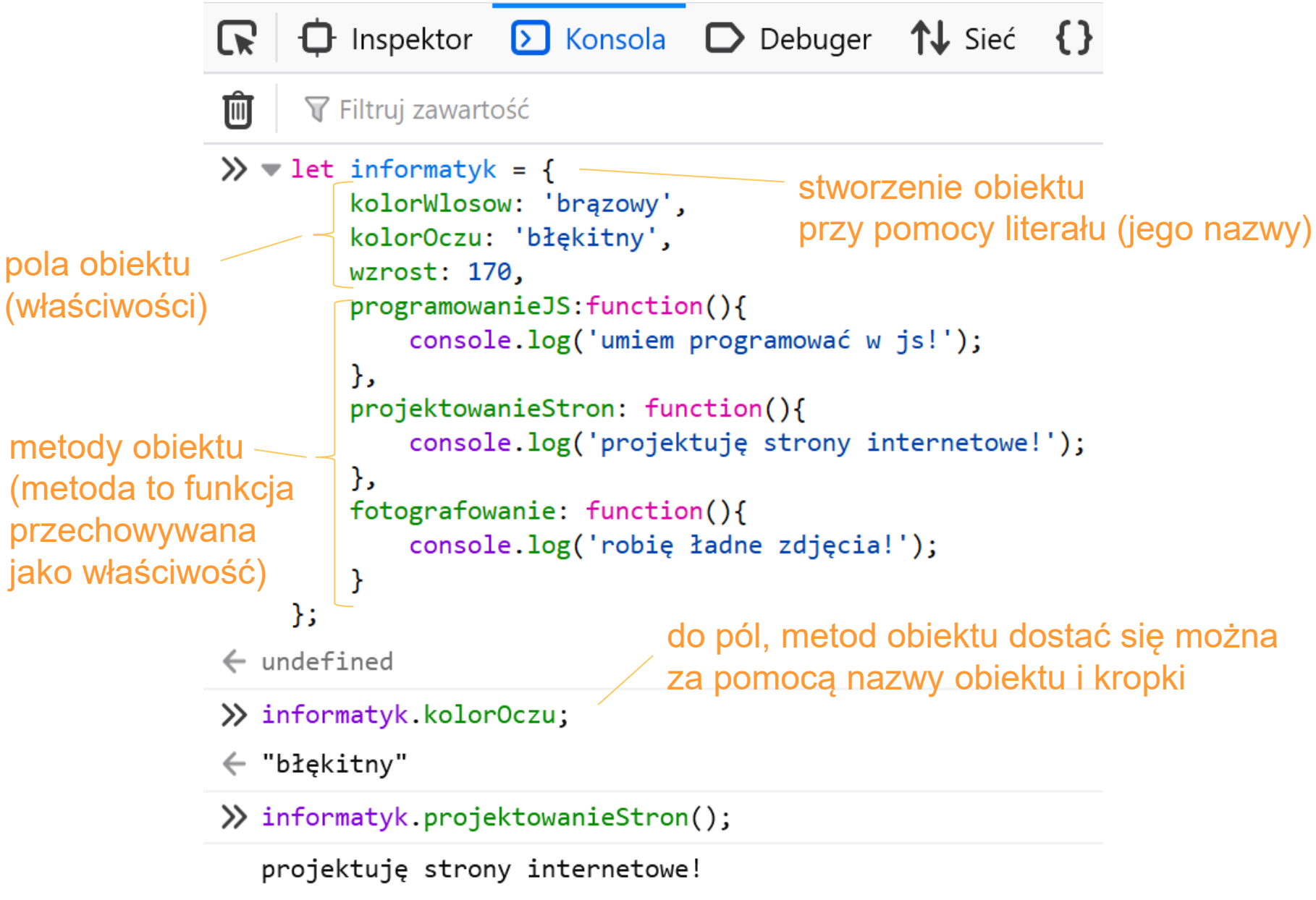
właściwości (pola) obiektu:
kolorWlosow: „brązowy”
kolorOczu: „błękitny”
wzrost: 170

Methods

metody obiektu:
programowanieJS()
projektowanieStron()
fotografowanie()

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

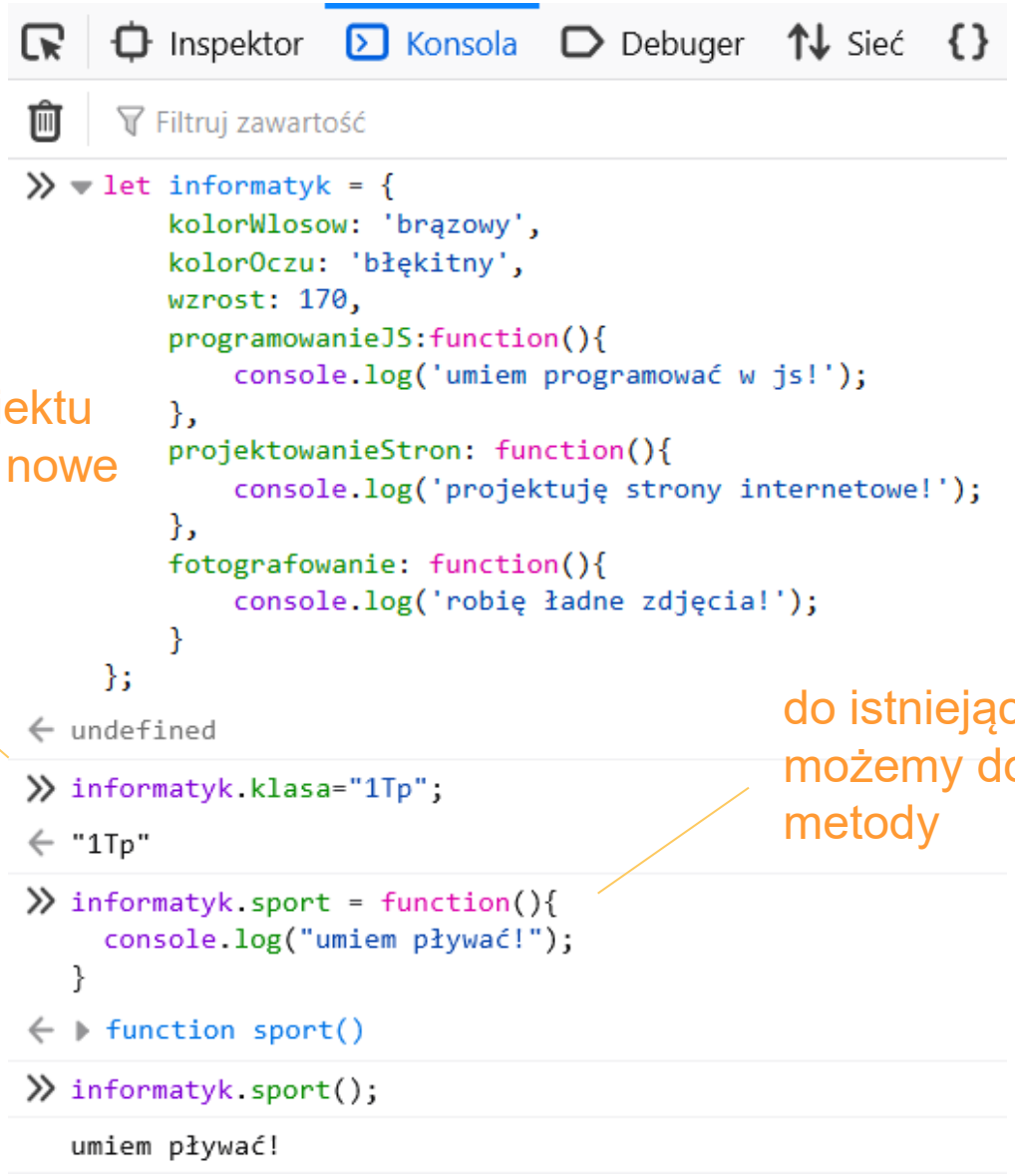
Tworzenie obiektu za pomocą literału



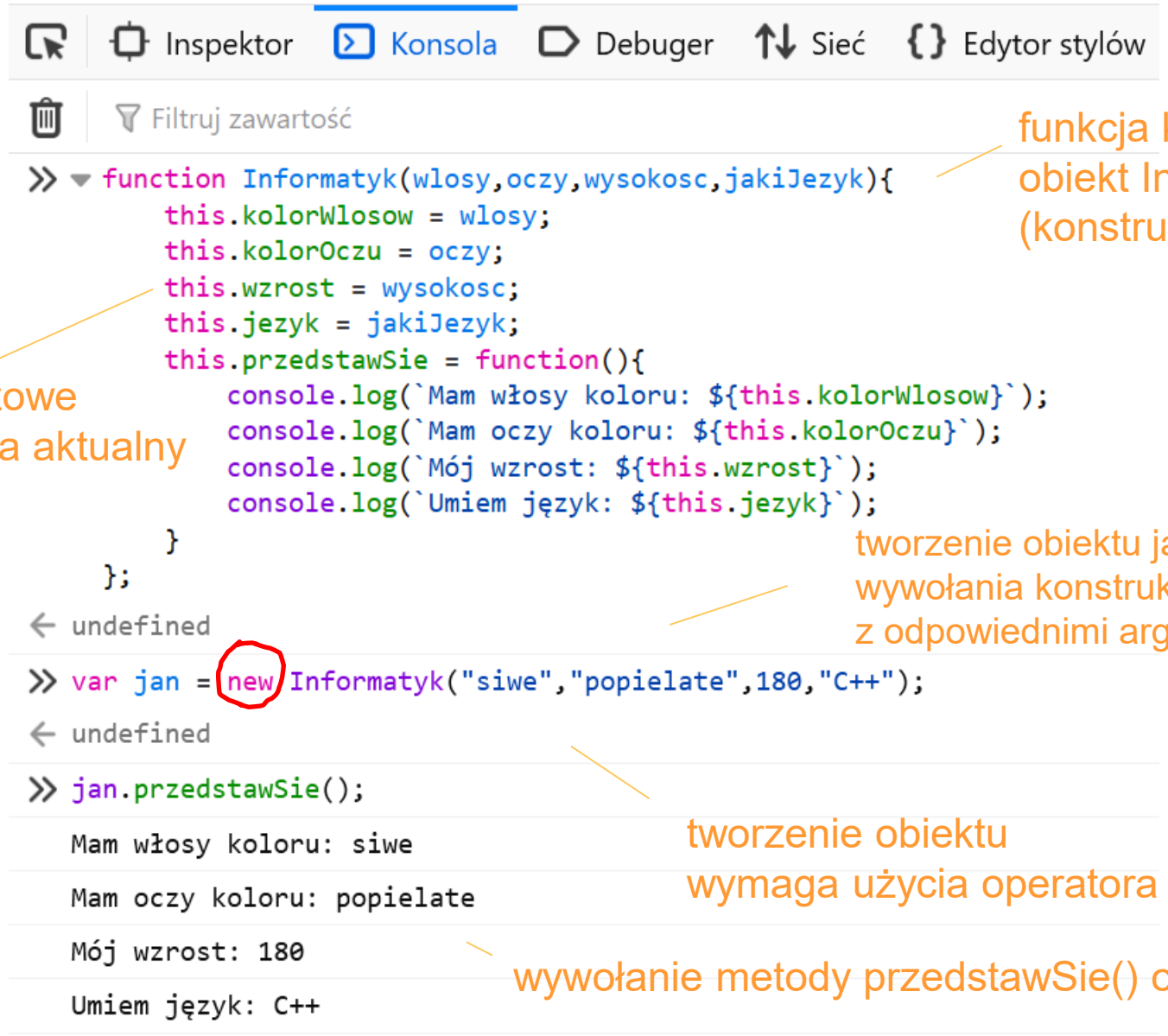
dodanie nowych pól i metod do istniejącego obiektu

do istniejącego obiektu możemy dodawać nowe pola

do istniejącego obiektu możemy dodawać nowe metody



Tworzenie obiektu za pomocą funkcji konstruującej obiekt



funkcja konstruująca obiekt Informatyk (konstruktor)

słowo kluczowe this oznacza aktualny obiekt

tworzenie obiektu jan wymaga wywołania konstruktora z odpowiednimi argumentami

tworzenie obiektu wymaga użycia operatora new

wywołanie metody przedstawSie() obiektu jan

dodanie nowych pól i metod do obiektu

do konstruktora można dodać pole
albo metodę za pomocą pola prototype

dodanie pola klasa

```
>> Informatyk.prototype.klasa = "1Tp";
<- "1Tp"
>> let mirek = new Informatyk("czarne","piwne",200,"Java");
<- undefined
```

sprawdzenie nowego pola klasa

```
>> mirek.klasa
<- "1Tp"
```

dodanie nowej metody

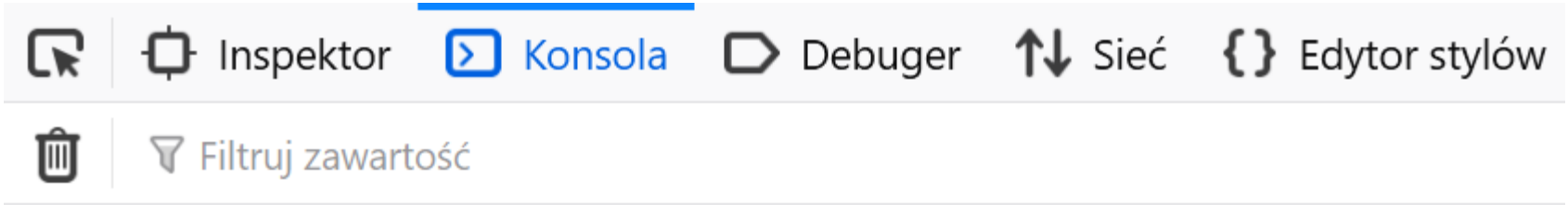
```
>> Informatyk.prototype.podajWzrost = function(){
     console.log(`mój wzrost: ${this.wzrost}`);
   };
<- ▶ function podajWzrost()
```

wywołanie nowej metody

```
>> mirek.podajWzrost();
   mój wzrost: 200
```

Klasa

złożony typ danych
(template obiektu)




```
class Informatyk {
    kolorWlosow;
    kolorOczu;
    wzrost;
    jezyk;
    constructor(wlosy,oczy,wysokosc,jakiJezyk){
        this.kolorWlosow = wlosy;
        this.kolorOczu = oczy;
        this.wzrost = wysokosc;
        this.jezyk = jakiJezyk;
    }
    przedstawSie(){
        console.log(`Mam włosy koloru: ${this.kolorWlosow}`);
        console.log(`Mam oczy koloru: ${this.kolorOczu}`);
        console.log(`Mój wzrost: ${this.wzrost}`);
        console.log(`Umiem język: ${this.jezyk}`);
    }
};
```

pola klasy

konstruktor klasy

konstruktor
tworzy obiekt
klasy Informatyk

metoda klasy

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes

Tworzenie obiektu za pomocą klasy

nazwa tworzonego obiektu

wywołanie parametrowego konstruktora klasy

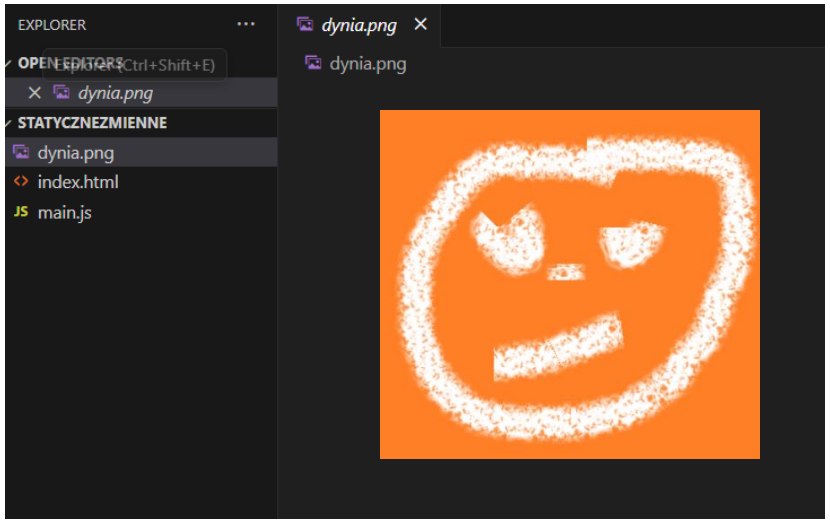
```
» const ania = new Informatyk("brązowe","niebieskie",170,"JavaScript");
← undefined
» ania.przedstawSie();
   Mam włosy koloru: brązowe
   Mam oczy koloru: niebieskie
   Mój wzrost: 170
   Umiem język: JavaScript
```

operator *new*

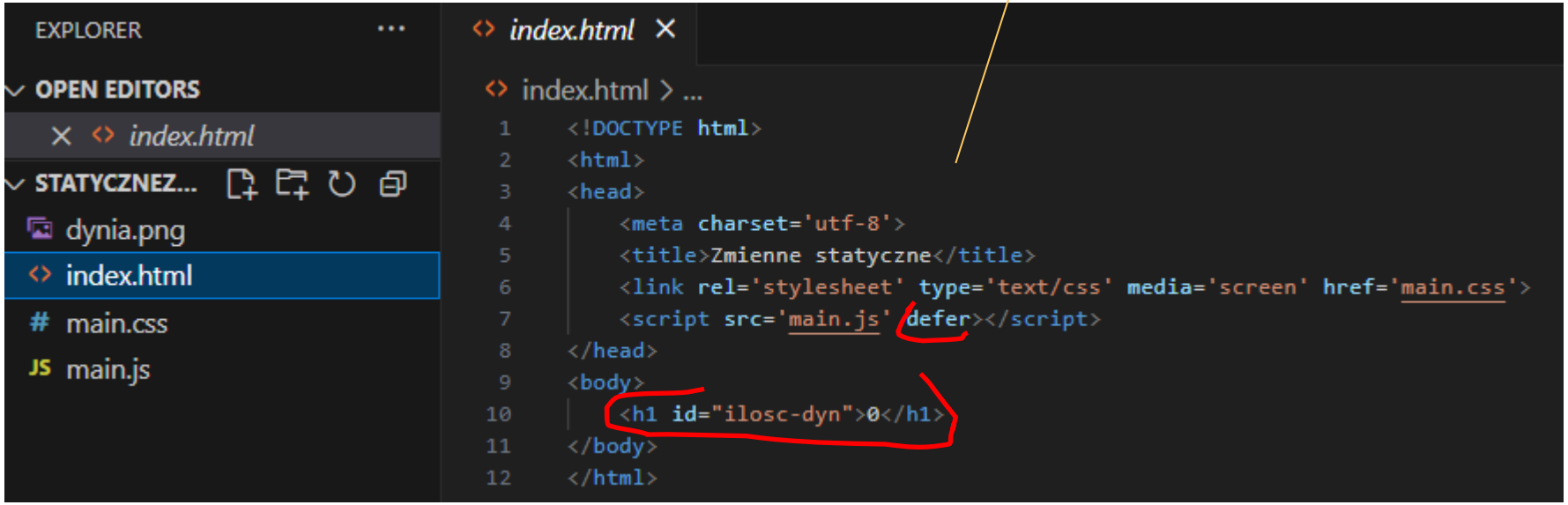
argumenty, które wypełnią pola obiektu

wywołanie metody przedstawSie() obiektu ania

Metody i pola statyczne klasy



defer - skrypt uruchomi się po załadowaniu strony



Metody i pola statyczne klasy

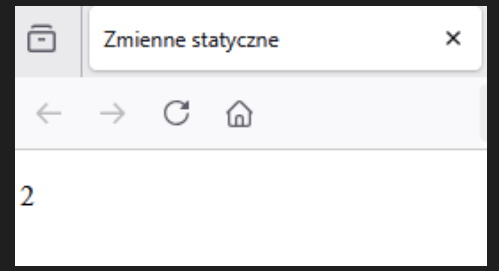
EXPLORER

OPEN EDITORS
- main.js

STATYCZNEZMIENNE
- dynia.png
- index.html
- main.js

main.js

```
let iloscDyn = document.getElementById("ilosc-dyn");

/**
 * Pole statyczne to takie, które należy do klasy, a nie do obiektu (instancji) tej klasy.
 *  - jest wspólne dla wszystkich obiektów,
 *  - nie trzeba tworzyć instancji, żeby go użyć,
 *  - dostęp uzyskujemy przez nazwę klasy
 */

class Halloween {
    static ilosc = 0;   // ilość stworzonych obiektów klasy Halloween
    constructor() {
        Halloween.ilosc++; // do pola statycznego dostajemy się po nazwie klasy
    }
}

let dynia1 = new Halloween();
let dynia2 = new Halloween();

iloscDyn.innerText = Halloween.ilosc;  // 2
```



Metody i pola statyczne klasy

EXPLORER

OPEN EDITORS
- main.js

STATYCZNEZMIENNE
- dynia.png
- index.html
- main.css
- main.js

main.js

```
let iloscDyn = document.getElementById("ilosc-dyn");

/**
 * Pole statyczne to takie, które należy do klasy, a nie do obiektu (instancji) tej klasy.
 *  - jest wspólne dla wszystkich obiektów,
 *  - nie trzeba tworzyć instancji, żeby go użyć,
 *  - dostęp uzyskujemy przez nazwę klasy
 */

class Halloween {
    static ilosc = 0;       // ilość stworzonych obiektów klasy Halloween
    constructor() {
        Halloween.ilosc++;  // do pola statycznego dostajemy się po nazwie klasy
        this.stworzDynie(); // wstawiamy obrazki na stronę
    }
    stworzDynie(){
        // tworzymy obrazek
        const img = document.createElement('img');
        img.src = 'dynia.png';
        img.alt = 'Halloween';

        // losowa pozycja
        const x = Math.random() * (window.innerWidth - 300);
        const y = Math.random() * (window.innerHeight - 300);
        img.style.left = `${x}px`;
        img.style.top = `${y}px`;

        // wstawiamy na stronę
        document.body.appendChild(img);

        // obrazek znika po podamym czasie
        setTimeout(() => img.remove(), 1000);
    }
}


// obrazek pojawia się co podany interwał
setInterval(()=>{
    new Halloween();
    iloscDyn.innerText = Halloween.ilosc;
}, 1);
```

EXPLORER

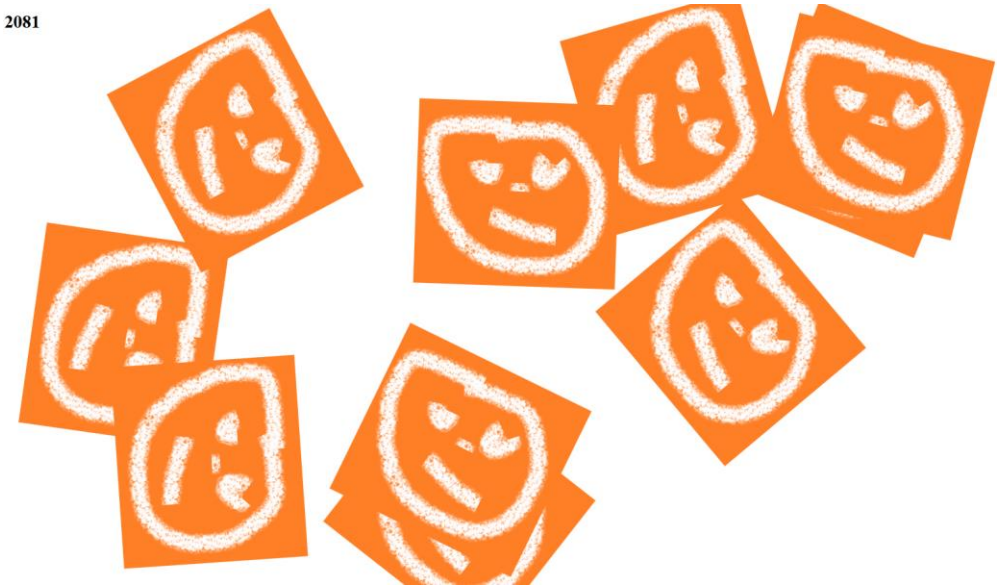
OPEN EDITORS
- main.css

STATYCZNEZ...
- dynia.png
- index.html
- main.css
- main.js

main.css

```
img{
    position: absolute;
    animation: spin 3s linear infinite; /* obracanie */
    transform-origin: center center; /* punkt obrotu */
}

@keyframes spin {           /* konfiguracja obrotu */
    from { transform: rotate(0deg); }
    to { transform: rotate(360deg); }
}

```

2081



Metody i pola statyczne klasy

pole statyczne klasy

metoda statyczna klasy

```
1   <!DOCTYPE html>
2   <html>
3   <body>
4
5   <p id="info"></p>
6
7   <script>
8   class MathematicalConstants {      // definicja klasy
9     static pi = 3.141592653589793;   // pole statyczne klasy
10    static podajStalaEulera(){       // metoda statyczna klasy
11      return 2.718281828459045;
12    }
13  }
14
15
16  document.getElementById("info").innerHTML =
17                "pi = " + MathematicalConstants.pi
18              + "<br>e = " + MathematicalConstants.podajStalaEulera()
19  </script>
20
21  </body>
22  </html>
```

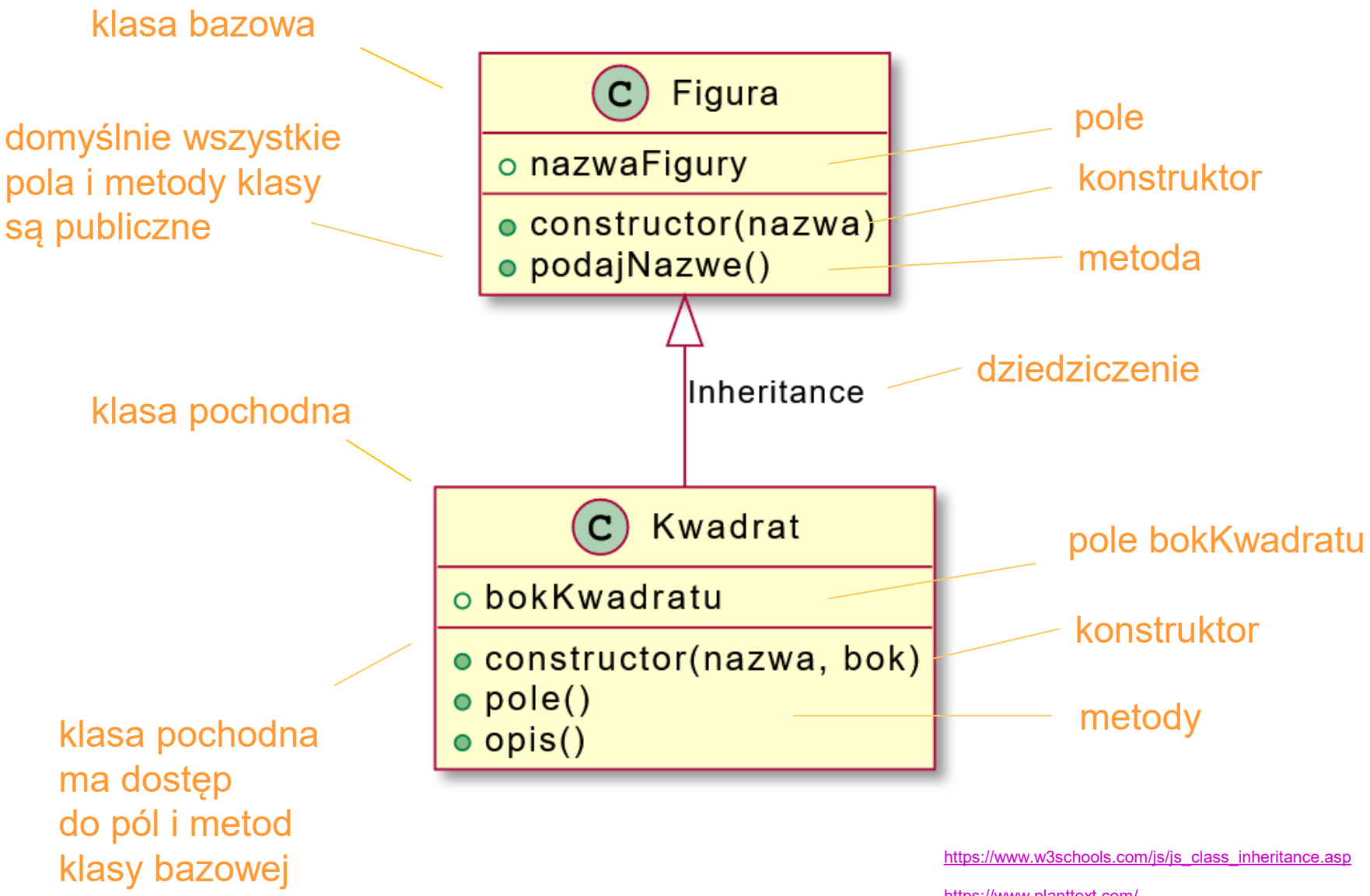
pola i metody statyczne wywołujemy poprzez nazwę klasy, a nie poprzez obiekt klasy

pi = 3.141592653589793
e = 2.718281828459045

podstawa logarytmu naturalnego

Dziedziczenie

Class Inheritance



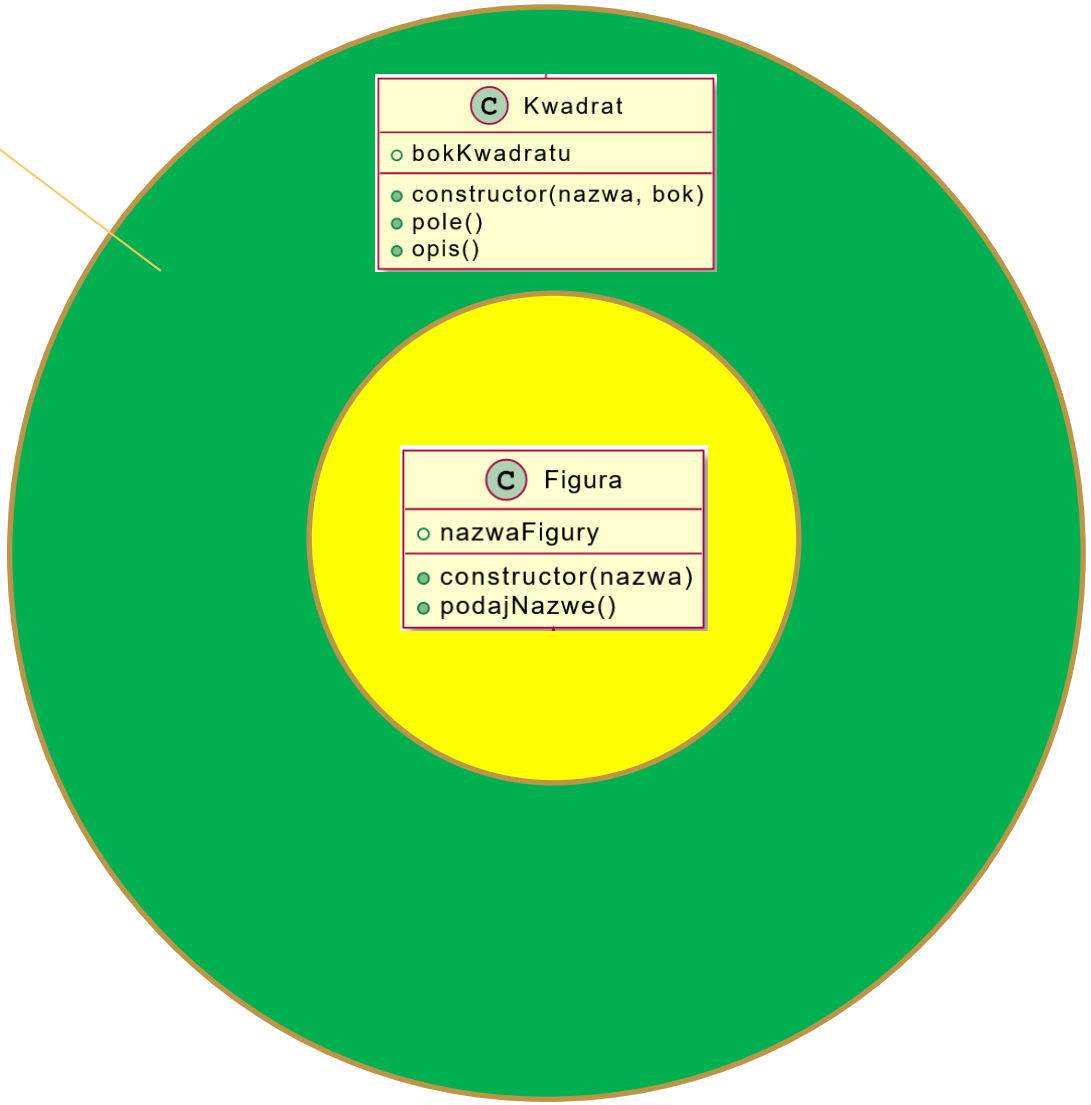
https://www.w3schools.com/js/js_class_inheritance.asp

https://www.planttext.com/

Dziedziczenie

Class Inheritance

obiekt klasy Kwadrat
zawiera w sobie
pola i metody klasy,
z której dziedziczy,
czyli klasy Figura



Dziedziczenie

Class Inheritance

```html
<!DOCTYPE html>
<html>
<body>

<p id="info"></p>

<script>
class Figura {                  // definicja klasy Figura
  nazwaFigury;                  // pole klasy
  constructor(nazwa) {          // konstruktor (tworzy obiekt klasy Figura)
    this.nazwaFigury = nazwa;   // przypisanie argumentu konstruktora (nazwa) do pola klasy (nazwaFigury)
  }
  podajNazwe() {                // metoda klasy wypisująca nazwę figury
    return 'Nazywam się ' + this.nazwaFigury;
  }
}

class Kwadrat extends Figura {  // klasa Kwadrat dziedziczy po klasie Figura
  bokKwadratu;                  // pole klasy
  constructor(nazwa, bok) {     // konstruktor (tworzy obiekt klasy Kwadrat)
    super(nazwa);               // wywołanie konstruktora klasy Figura i przekazanie mu nazwy figury
    this.bokKwadratu = bok;     // przypisanie argumentu konstruktora (bok) do pola klasy (bokKwadratu)
  }
  pole(){                       // metoda klasy obliczająca pole kwadratu
    return this.bokKwadratu * this.bokKwadratu;
  }
  opis() {                      // metoda klasy wypisująca dane o kwadracie
    return this.podajNazwe() + ", mój bok: " + this.bokKwadratu + ", pole: " + this.pole();
  }
}

const kwadrat = new Kwadrat("kwadrat", 20);   // utworzenie obiektu kwadrat klasy Kwadrat
document.getElementById("info").innerHTML = kwadrat.opis();  // wywołanie metody opis() obiektu kwadrat
</script>

</body>
</html>
```

klasa bazowa

klasa potomna dziedzicząca z klasy bazowej

klasa potomna (kwadrat), posiada dostęp do pól i metod klasy bazowej (Figura)

Nazywam się kwadrat, mój bok: 20, pole: 400

Obiekt String

String object

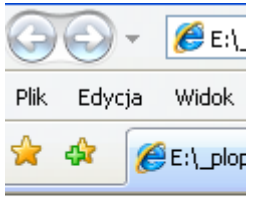
obiekt wbudowany języka JavaScript

operacje na łańcuchach znaków

deklaracja zmiennej *wyraz* typu string

właściwość obiektu

```
var wyraz="Informatyka";
document.write(wyraz.length);
```



11

wypisanie na stronie długości słowa *informatyka*

Obiekt String

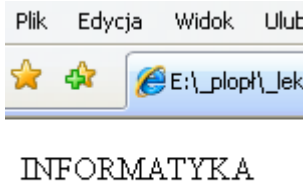
String object

operacje na łańcuchach znaków

deklaracja zmiennej *wyraz* typu string

metoda obiektu

```
var wyraz="Informatyka";
document.write(wyraz.toUpperCase() );
```



wypisanie na stronie słowa *INFORMATYKA*

String object

Obiekt String

String Object Properties

właściwości

Własciwość	**opis**
constructor	Referencja do konstruktora
length	Zwraca liczbę znaków w stringu
prototype	Pozwala dodawać właściwości i metody do obiektu

String references

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String

https://www.w3schools.com/jsref/jsref_obj_string.asp

String object

Obiekt String

Metody

String Object Methods

Metoda	Opis
anchor()	Creates an HTML anchor
big()	Displays a string in a big font
blink()	Displays a blinking string
bold()	Displays a string in bold
charAt()	Returns the character at a specified position
charCodeAt()	Returns the Unicode of the character at a specified position
concat()	Joins two or more strings
fixed()	Displays a string as teletype text
fontcolor()	Displays a string in a specified color
fontsize()	Displays a string in a specified size
fromCharCode()	Takes the specified Unicode values and returns a string
indexOf()	Returns the position of the first occurrence of a specified string value in a string

String object

Obiekt String

italics()	Displays a string in italic
lastIndexOf()	Returns the position of the last occurrence of a specified string value, searching backwards from the specified position in a string
link()	Displays a string as a hyperlink
match()	Searches for a specified value in a string
replace()	Replaces some characters with some other characters in a string
search()	Searches a string for a specified value
slice()	Extracts a part of a string and returns the extracted part in a new string
small()	Displays a string in a small font
split()	Splits a string into an array of strings
strike()	Displays a string with a strikethrough
sub()	Displays a string as subscript

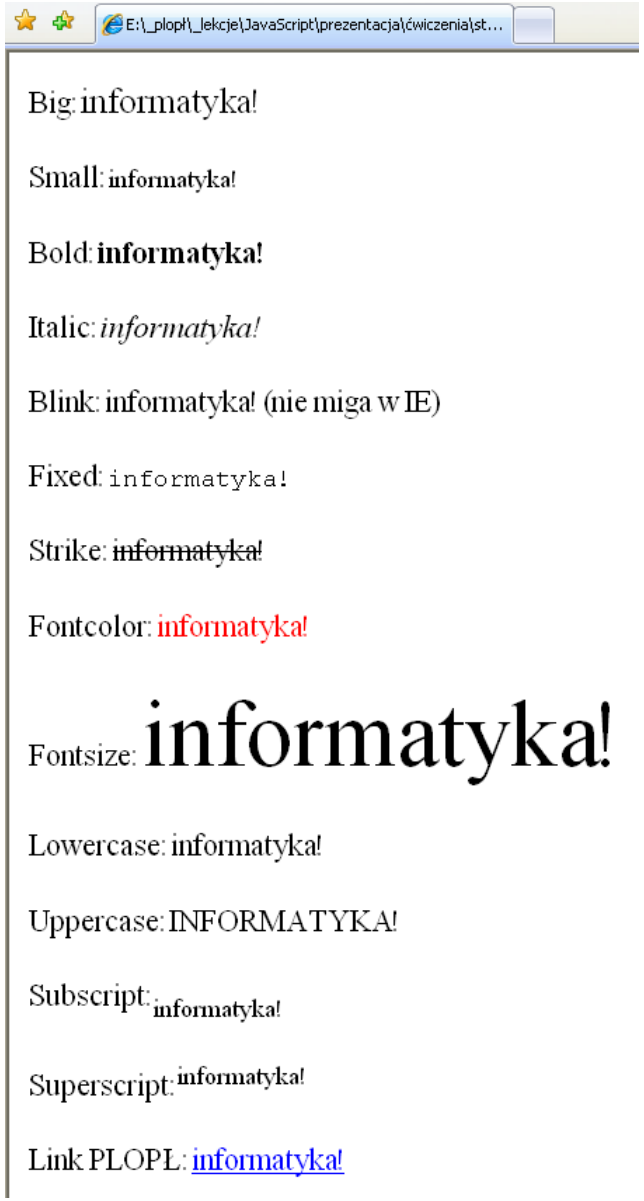
Obiekt String

String object

substr()	Extracts a specified number of characters in a string, from a start index
substring()	Extracts the characters in a string between two specified indices
sup()	Displays a string as superscript
toLowerCase()	Displays a string in lowercase letters
toUpperCase()	Displays a string in uppercase letters
toSource()	Represents the source code of an object
valueOf()	Returns the primitive value of a String object

Ćwiczenie

Napisać stronę wyświetlającą w różny sposób napis *informatyka.*



Ćwiczenia

1. Wypisz słowo *informatyka* za pomocą metody charAt()
2. Pobierz imię od użytkownika i wypisz je na stronie od końca.

Obiekt Math

umożliwia wykonywanie operacji matematycznych

Property	Description
E	Returns Euler's constant (approx. 2.718)
LN2	Returns the natural logarithm of 2 (approx. 0.693)
LN10	Returns the natural logarithm of 10 (approx. 2.302)
LOG2E	Returns the base-2 logarithm of E (approx. 1.442)
LOG10E	Returns the base-10 logarithm of E (approx. 0.434)
PI	Returns PI (approx. 3.14159)
SQRT1_2	Returns the square root of 1/2 (approx. 0.707)
SQRT2	Returns the square root of 2 (approx. 1.414)

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

Obiekt Math

Method	Description
abs(x)	Returns the absolute value of a number
acos(x)	Returns the arccosine of a number
asin(x)	Returns the arcsine of a number
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the angle theta of an (x,y) point as a numeric value between -PI and PI radians
ceil(x)	Returns the value of a number rounded upwards to the nearest integer
cos(x)	Returns the cosine of a number
exp(x)	Returns the value of E^x
floor(x)	Returns the value of a number rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of a number
max(x,y)	Returns the number with the highest value of x and y
min(x,y)	Returns the number with the lowest value of x and y
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds a number to the nearest integer
sin(x)	Returns the sine of a number
sqrt(x)	Returns the square root of a number
tan(x)	Returns the tangent of an angle
toSource()	Represents the source code of an object
valueOf()	Returns the primitive value of a Math object

Ćwiczenia

pi.html
1. Wypisz na stronie liczbę π .
2. Sprawdź równanie $\sqrt{2} = 1.414213562\ldots$

abs.html
2. Co oznacza skrót *abs?* Napisz skrypt wykorzystujący metodę abs().
3. Wypisz na stronie 10 liczb losowych jedna pod drugą.

random.html
4. Wypisz na stronie zadaną przez użytkownika ilość liczb losowych.
5. Wypisz na stronie 10 liczb losowych w z zakresu <0,100>

random1.html
6. Pobierz od użytkownika :
   - liczbę liczb losowych do wypisania
   - szukaną liczbę

   Wypisz na stronie zadaną przez użytkownika ilość liczb losowych oraz podaj, ile razy pojawiła się zadana przez użytkownika szukana liczba.

Ćwiczenia

7. Napisz skrypt realizujący losowanie:
   - jednej liczby
   - dwóch liczb
   - trzech liczb

   w zakresie <1,49>

losowanie_jednej_liczby.html

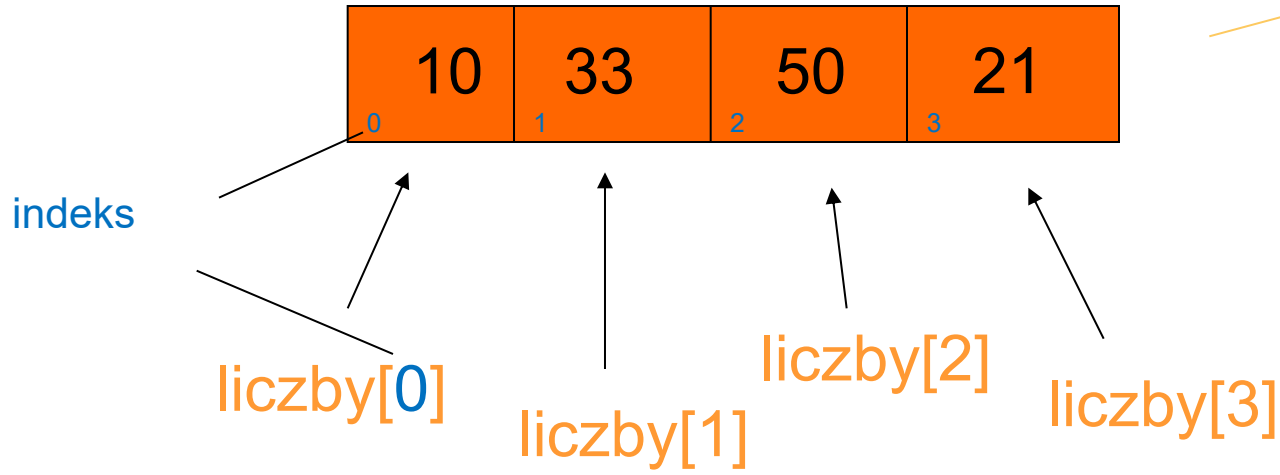
losowanie_dwoch_liczb.html

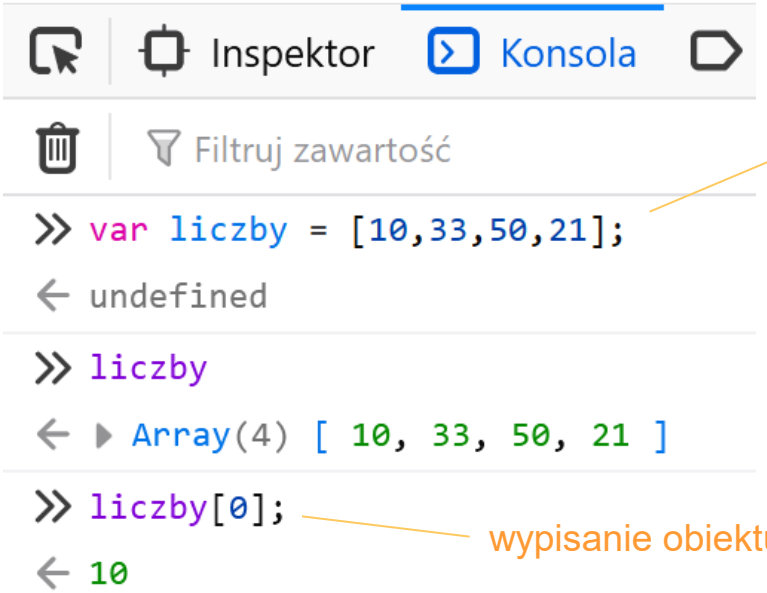
losowanie_trzech_liczb.html

Obiekt Array

tablica do przechowywania danych

tablica

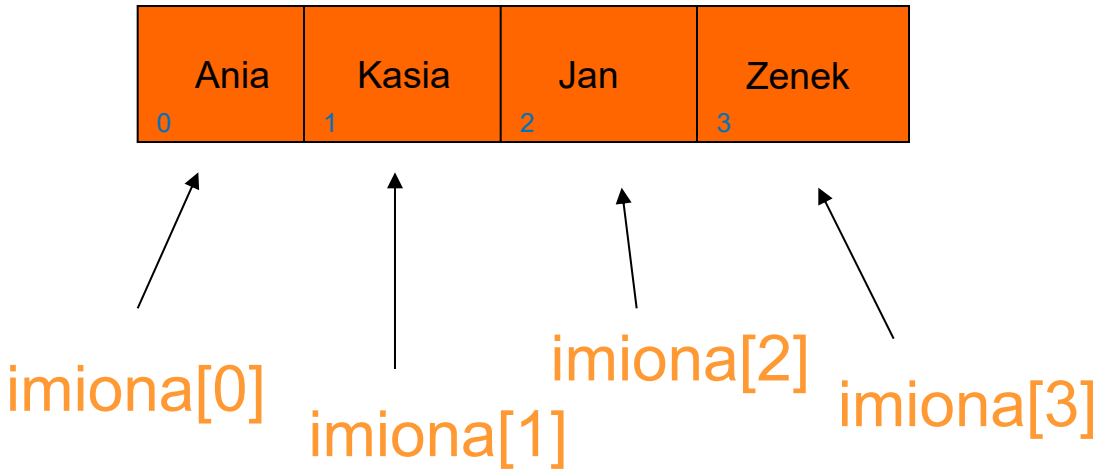


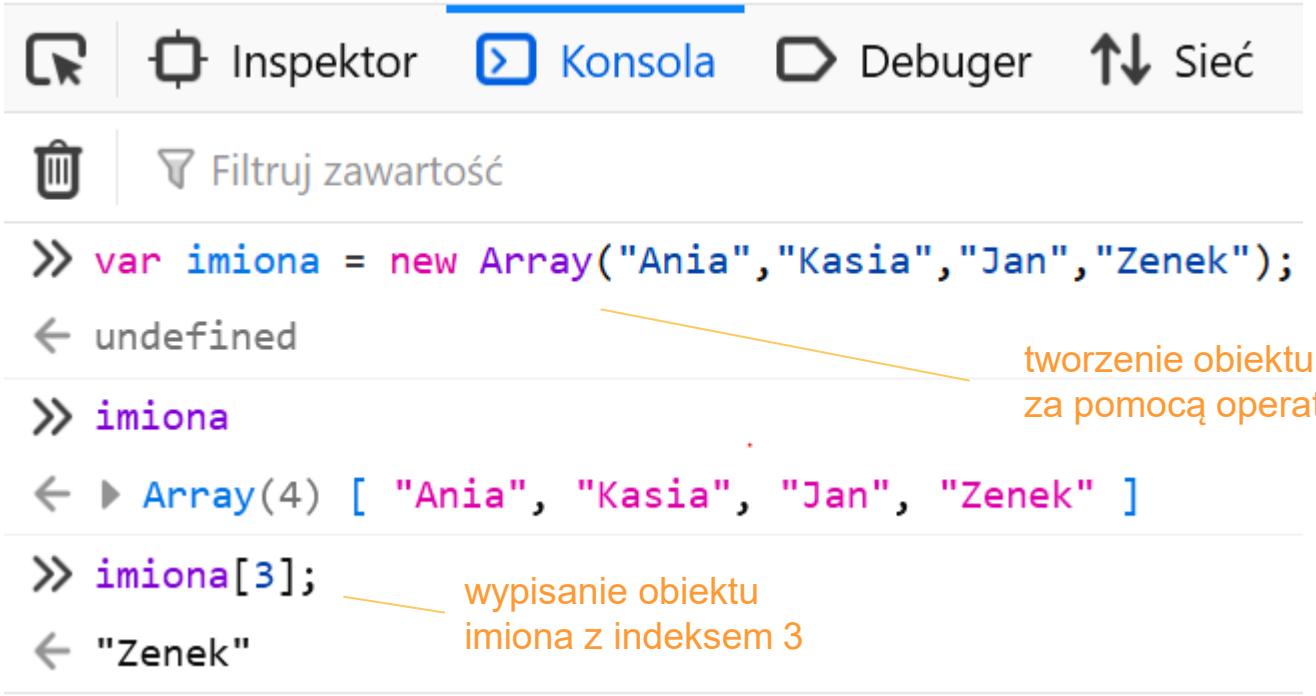


tworzenie obiektu Array (tablicy)

wypisanie obiektu liczby z indeksem 1

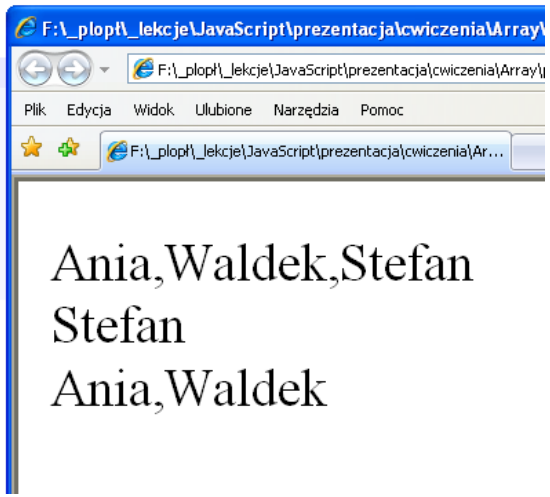
Obiekt Array





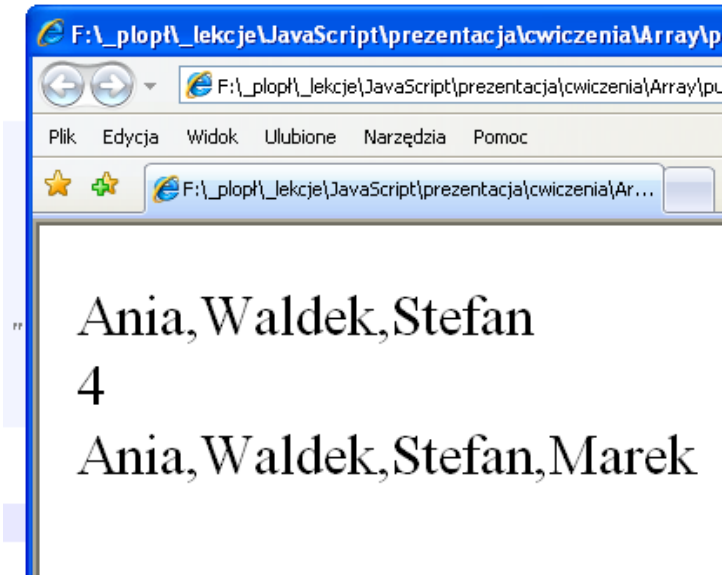
Array - metoda pop()

```
<script>

var imiona = new Array(3);
 imiona[0] = „Ania";
 imiona[1] = „Waldek";
 imiona[2] = „Stefan";

document.write(imiona + "<br>");
document.write(imiona.pop() + "<br>");
document.write(imiona);

</script>
```



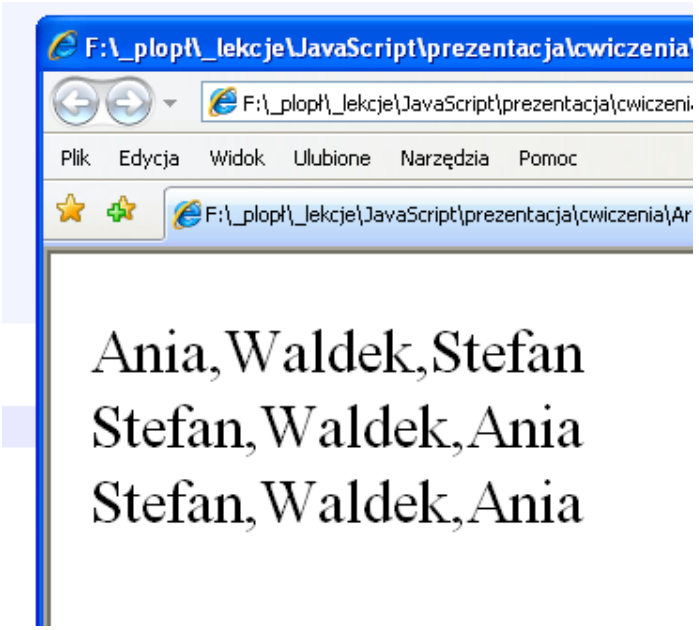
Array - metoda push()

```
<script>

var imiona = new Array(3);
imiona[0] = "Ania";
imiona[1] = "Waldek";
imiona[2] = "Stefan";
document.write(imiona + "<br>");
document.write(imiona.push("Marek") + "<br>");
document.write(imiona);

</script>
```



Array - metoda reverse()

```
<script>

var imiona = new Array(3);
imiona[0] = "Ania";
imiona[1] = "Waldek";
imiona[2] = "Stefan";
document.write(imiona + "<br>");
document.write(imiona.reverse() + "<br>");
document.write(imiona);

</script>
```



Obiekt Array

Property	Description
constructor	Returns a reference to the array function that created the object
index	
input	
length	Sets or returns the number of elements in an array
prototype	Allows you to add properties and methods to the object

Array references

https://www.w3schools.com/jsref/jsref_obj_array.asp

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Obiekt Array

Method	Description
concat()	Joins two or more arrays and returns the result
join()	Puts all the elements of an array into a string. The elements are separated by a specified delimiter
pop()	Removes and returns the last element of an array
push()	Adds one or more elements to the end of an array and returns the new length
reverse()	Reverses the order of the elements in an array
shift()	Removes and returns the first element of an array
slice()	Returns selected elements from an existing array
sort()	Sorts the elements of an array
splice()	Removes and adds new elements to an array
toSource()	Represents the source code of an object
toString()	Converts an array to a string and returns the result
unshift()	Adds one or more elements to the beginning of an array and returns the new length
valueOf()	Returns the primitive value of an Array object

Obiekt Array

Ania Waldek Stefan

pop push sort revers

napisz skrypt ilustrujący działanie metod:

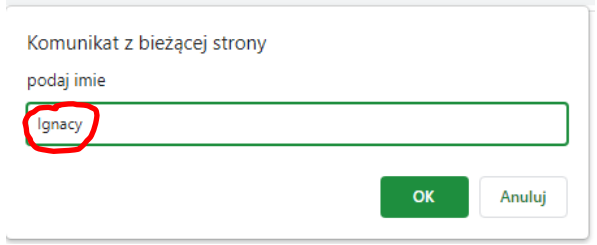
- pop
- push
- sort
- revers

Ania Waldek

pop push sort revers

Ania Waldek

pop push sort revers



Ania Waldek Ignacy

pop push sort revers

imiona są zapisane w tablicy

Ania Ignacy Waldek

pop push sort revers

Waldek Ignacy Ania

pop push sort revers

Obiekt Array

```html
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<h1 id="info"></h1>
<button id="imiona-pop">pop</button>
<button id="imiona-push">push</button>
<button id="imiona-sort">sort</button>
<button id="imiona-reverse">revers</button>
<script>
let info = document.getElementById("info");
let imionaPop = document.getElementById("imiona-pop");
let imionaPush = document.getElementById("imiona-push");
let imionaSort = document.getElementById("imiona-sort");
let imionaReverse = document.getElementById("imiona-reverse");
let napis = "";

var imiona = new Array(3);
imiona[0] = "Ania";
imiona[1] = "Waldek";
imiona[2] = "Stefan";

function wypisz(){
    napis = "";
    for (var i = 0; i < imiona.length; i++) {
        napis += imiona[i] + " ";
    }
    info.innerText = napis;
}

imionaPop.onclick = function(){
    imiona.pop();
    wypisz();
}

imionaPush.onclick = function(){
    let nowe = prompt("podaj imie");
    imiona.push(nowe);
    wypisz();
}

imionaSort.onclick = function(){
    imiona.sort();
    wypisz();
}

imionaReverse.onclick = function(){
    imiona.reverse();
    wypisz();
}

wypisz();

</script>
</body>
</html>
```

przyciski na stronie

identyfikatory przycisków

tablica z imionami

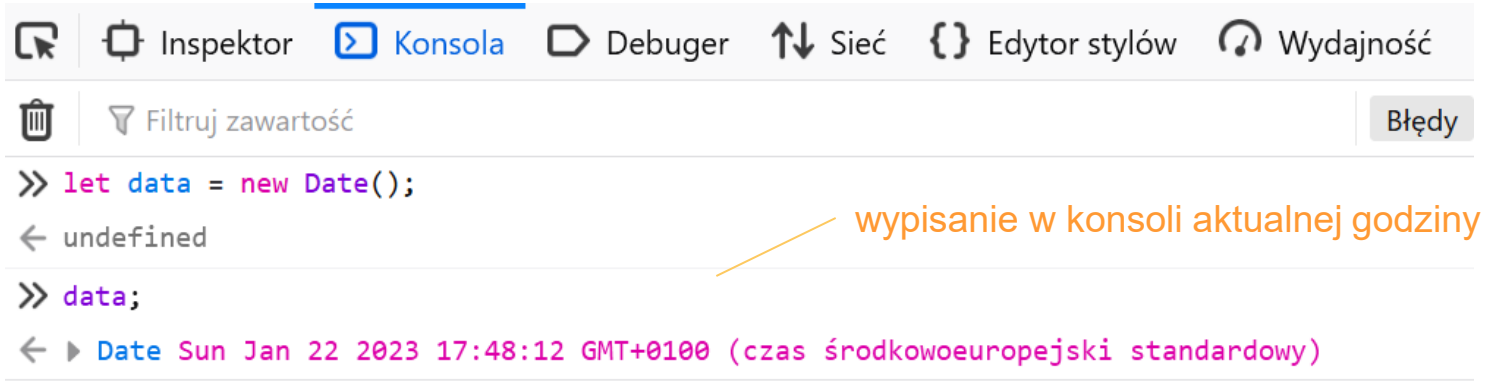
funkcja wypisująca imiona z tablicy

funkcje obsługujące przyciski

jednokrotnie wypisujemy imiona przy starcie strony

Obiekt Date

przechowuje datę i czas



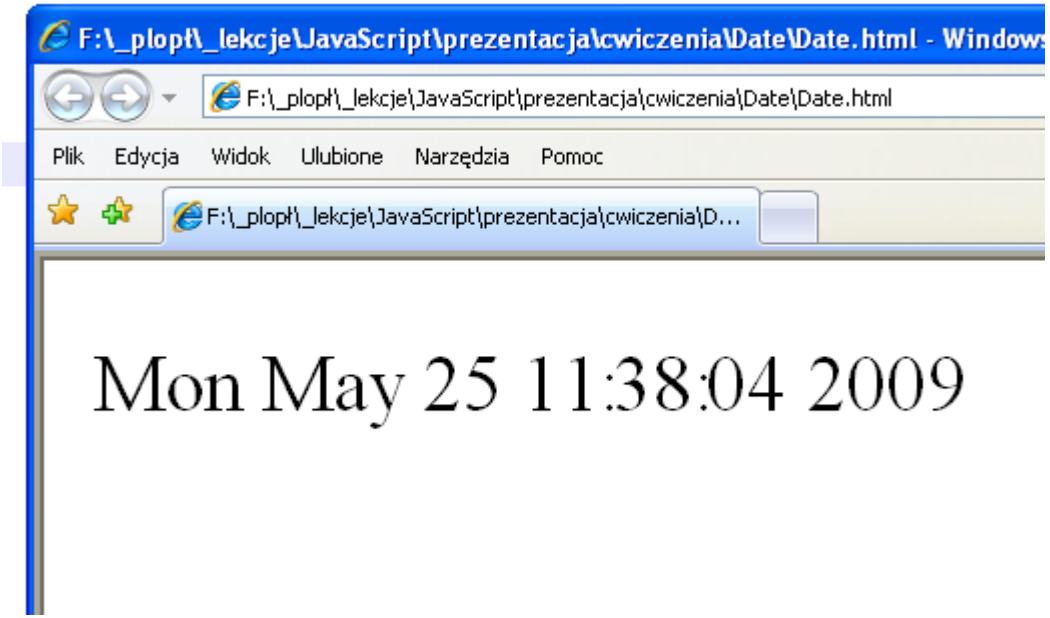
wypisanie w konsoli aktualnej godziny i czasu

Property	Description
constructor	Returns a reference to the Date function that created the object
prototype	Allows you to add properties and methods to the object

https://www.w3schools.com/jsref/jsref_obj_date.asp

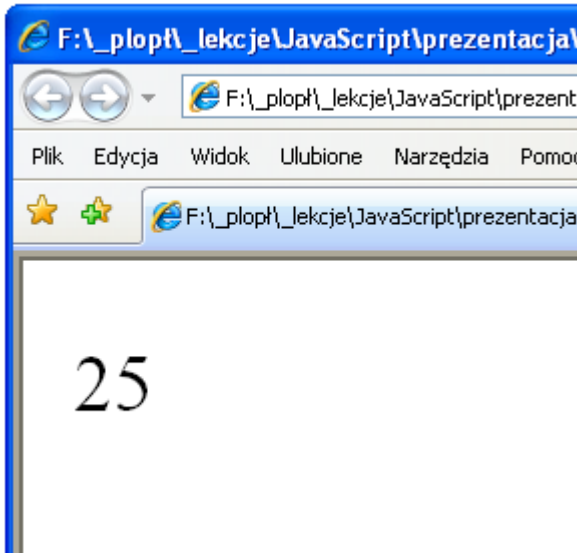
Date – metoda Date()

```
<script>

document.write(Date());

</script>
```



Date – metoda getDate()

```
<script>

var d = new Date();
document.write(d.getDate());

</script>
```



Obiekt Date

Method	Description
Date()	Returns today's date and time
getDate()	Returns the day of the month from a Date object (from 1-31)
getDay()	Returns the day of the week from a Date object (from 0-6)
getFullYear()	Returns the year, as a four-digit number, from a Date object
getHours()	Returns the hour of a Date object (from 0-23)
getMilliseconds()	Returns the milliseconds of a Date object (from 0-999)
getMinutes()	Returns the minutes of a Date object (from 0-59)
getMonth()	Returns the month from a Date object (from 0-11)
getSeconds()	Returns the seconds of a Date object (from 0-59)
getTime()	Returns the number of milliseconds since midnight Jan 1, 1970
getTimezoneOffset()	Returns the difference in minutes between local time and Greenwich Mean Time (GMT)
getUTCDate()	Returns the day of the month from a Date object according to universal time (from 1-31)
getUTCDay()	Returns the day of the week from a Date object according to universal time (from 0-6)
getUTCMonth()	Returns the month from a Date object according to universal time (from 0-11)
getUTCFullYear()	Returns the four-digit year from a Date object according to universal time
getUTCHours()	Returns the hour of a Date object according to universal time (from 0-23)
getUTCMinutes()	Returns the minutes of a Date object according to universal time (from 0-59)
getUTCSeconds()	Returns the seconds of a Date object according to universal time (from 0-59)
getUTCMilliseconds()	Returns the milliseconds of a Date object according to universal time (from 0-999)
getYear()	Returns the year, as a two-digit or a three/four-digit number, depending on the browser. Use getFullYear() instead !!

Obiekt Date

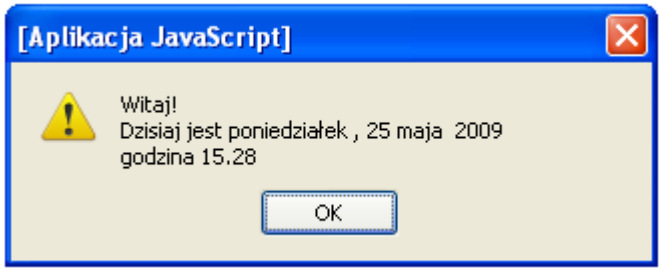
parse()	Takes a date string and returns the number of milliseconds since midnight of January 1, 1970
setDate()	Sets the day of the month in a Date object (from 1-31)
setFullYear()	Sets the year in a Date object (four digits)
setHours()	Sets the hour in a Date object (from 0-23)
setMilliseconds()	Sets the milliseconds in a Date object (from 0-999)
setMinutes()	Set the minutes in a Date object (from 0-59)
setMonth()	Sets the month in a Date object (from 0-11)
setSeconds()	Sets the seconds in a Date object (from 0-59)
setTime()	Calculates a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970
setUTCDate()	Sets the day of the month in a Date object according to universal time (from 1-31)
setUTCMonth()	Sets the month in a Date object according to universal time (from 0-11)
setUTCFullYear()	Sets the year in a Date object according to universal time (four digits)
setUTCHours()	Sets the hour in a Date object according to universal time (from 0-23)
setUTCMinutes()	Set the minutes in a Date object according to universal time (from 0-59)
setUTCSeconds()	Set the seconds in a Date object according to universal time (from 0-59)
setUTCMilliseconds()	Sets the milliseconds in a Date object according to universal time (from 0-999)
setYear()	Sets the year in the Date object (two or four digits). Use setFullYear() instead !!
toDateString()	Returns the date portion of a Date object in readable form
toGMTString()	Converts a Date object, according to Greenwich time, to a string. Use toUTCString() instead !!
toLocaleDateString()	Converts a Date object, according to local time, to a string and returns the date portion
toLocaleTimeString()	Converts a Date object, according to local time, to a string and returns the time portion

Obiekt Date

toLocaleString()	Converts a Date object, according to local time, to a string
toSource()	Represents the source code of an object
toString()	Converts a Date object to a string
toTimeString()	Returns the time portion of a Date object in readable form
toUTCString()	Converts a Date object, according to universal time, to a string
UTC()	Takes a date and returns the number of milliseconds since midnight of January 1, 1970 according to universal time
valueOf()	Returns the primitive value of a Date object

Ćwiczenia

1. Napisz skrypt wyświetlający bieżącą datę w formacie:



uwaga:

data.getDay() zwaraca cyfry od 0 do 6
dlatego można wartość zwracaną traktować jak indeks odpowiedniej tablicy
np.:

```
dzien_tygodnia = new Array(7)
dzien_tygodnia[0] = "niedziela "
dzien_tygodnia[1] = "poniedziałek "
…..

let dzien = dzien_tygodnia[data.getDay()]
```

tablica z polskimi nazwami dni tygodnia

aktualny dzień tygodnia po polsku

DOM

Document Object Model

DOM

Standard organizacji W3C

HTML DOM

HTML DOM to model obiektowy dla HTML

Elementy HTML jako **obiekty**

Metody dla wszystkich elementów HTML

DOM definiuje

Właściwości wszystkich elementów HTML

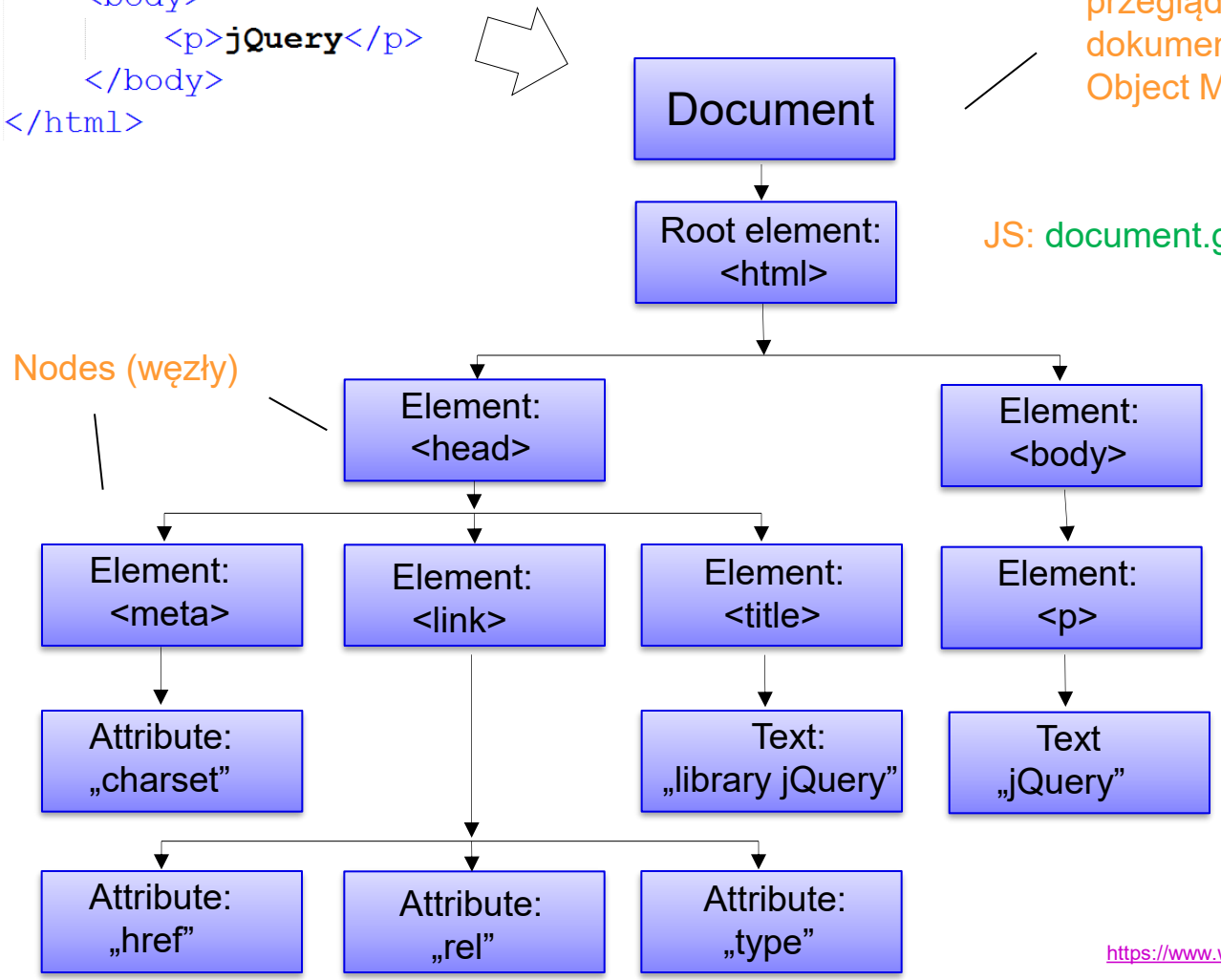
Zdarzenia dla wszystkich elementów HTML

https://www.w3schools.com/whatis/whatis_htmldom.asp

DOM HTML

```
<html>
    <head>
        <meta charset="utf-8">
        <link href="style.css" rel="stylesheet" type="text/css">
        <title>library jQuery</title>
    </head>
    <body>
        <p>jQuery</p>
    </body>
</html>
```

Po załadowaniu strony internetowej przeglądarka tworzy obiektowy model dokumentu strony HTML DOM (Document Object Model)



Nodes (węzły)

JS: document.getElementsByTagName("p")[0];

dostęp do węzła <p>

HTML DOM to API (Application Programming Interface) dla JavaScript

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

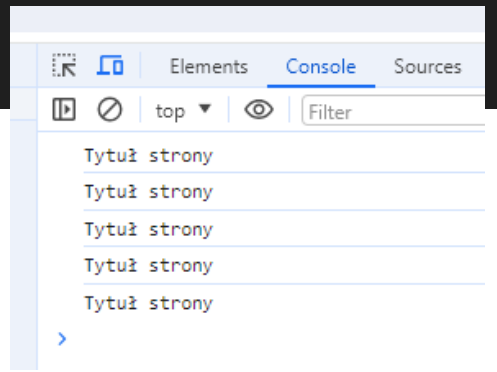
DOM HTML

HTML DOM to API (Application Programming Interface) dla JavaScript

index.html > ...

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <meta charset='utf-8'>
5       <title>Page Title</title>
6   </head>
7   <body>
8       <h1 class="classTitle" id="idTitle" name="nameTitle">Tytuł strony</h1>
9       <script>
10          console.log(document.getElementsByTagName("h1")[0].innerText);
11          console.log(document.getElementById("idTitle").innerText);
12          console.log(document.getElementsByClassName("classTitle")[0].innerText);
13          console.log(document.getElementsByName("nameTitle")[0].innerText);
14          console.log(document.querySelector("h1").innerText);
15      </script>
16  </body>
17  </html>
```

różne sposoby dostania się do treści elementu h1 na stronie poprzez js

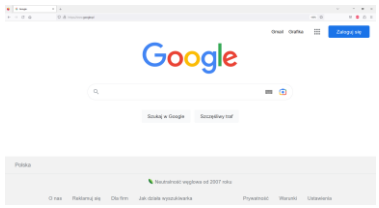


wybrane obiekty strony

każda strona internetowa
jest związana z wieloma obiektami

window

location



nawigator

history

document

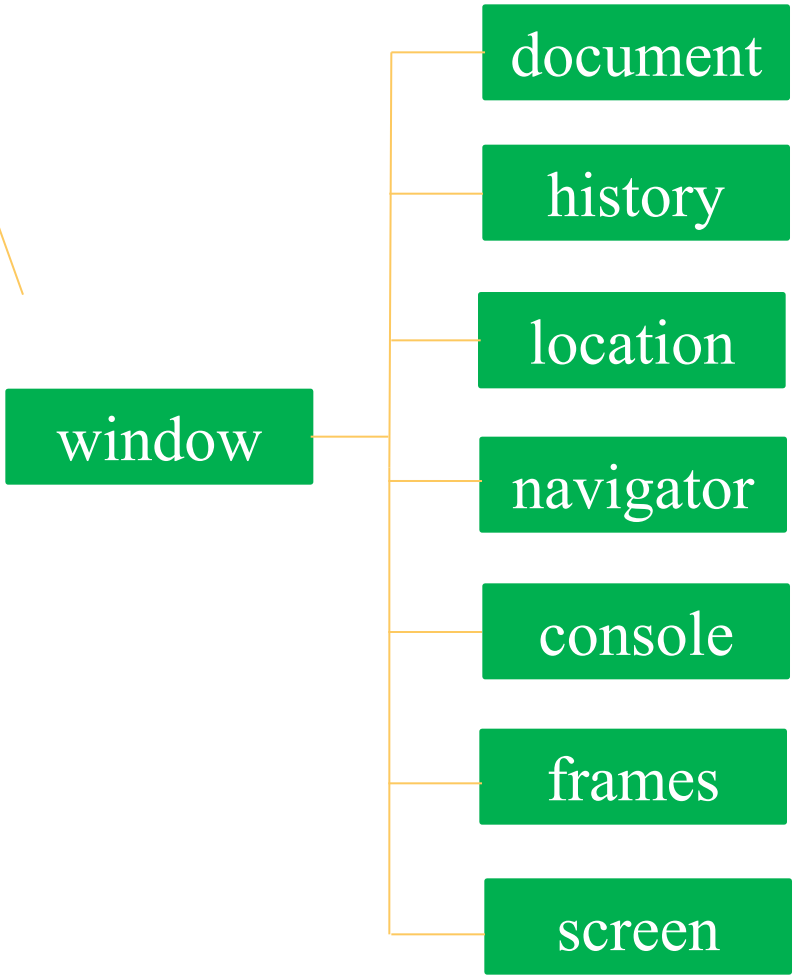
przykładowe obiekty strony

https://www.w3schools.com/jsref/default.asp

obiekt window

obiekt window jest nadrzędny w stosunku do innych obiektów strony

reprezentuje okno przeglądarki

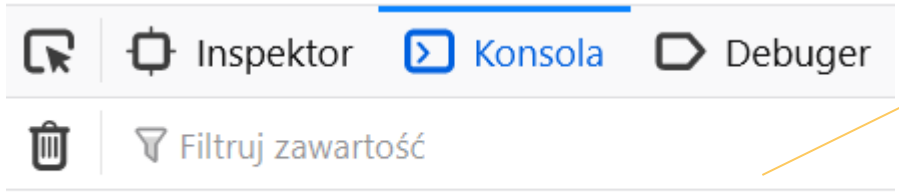


pola obiektu window, które zawierają inne obiekty

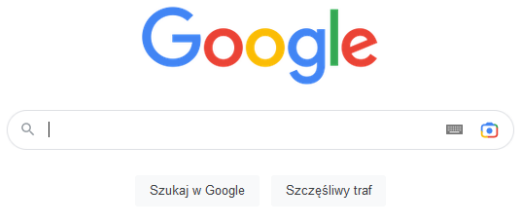
https://www.w3schools.com/jsref/obj_window.asp

obiekt window

otwarcie nowego okna




```
window.open("http://www.google.pl");
```




```
window.innerHeight
25
```

wysokość okna przeglądarki

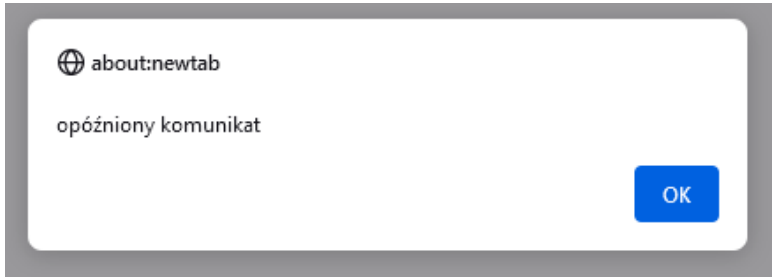
```
window.innerWidth
1920
```

szerokość okna przeglądarki

```
showAlert = function(){
  alert("opóźniony komunikat");
};
function showAlert()
```

opóźnienie

```
window.setTimeout(showAlert,5000);
2
```



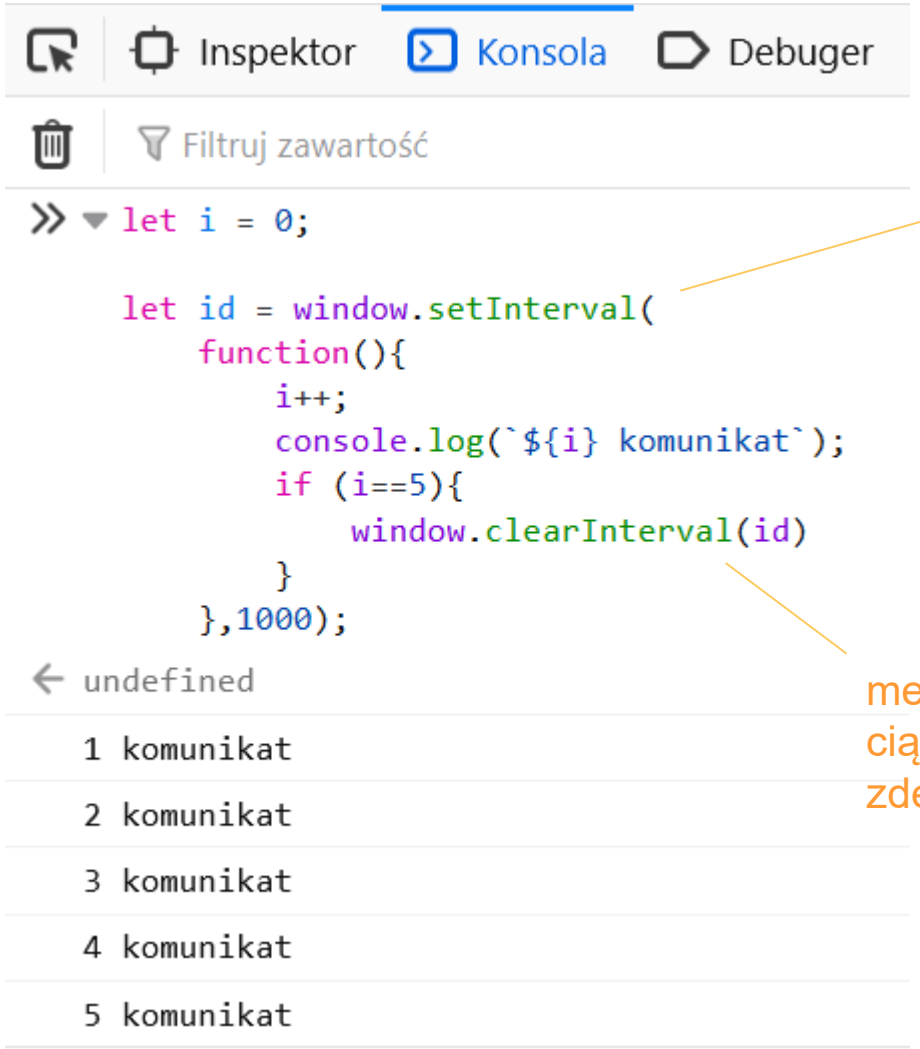
opóźnienie wywołania funkcji showAlert o 5 s,
funkcja wykona się jednokrotnie

okienko komunikatu pojawia się po 5s od uruchomienia kodu

obiekt window

metoda setInterval wywołuje funkcję showAlert co 1s (w nieskończoność)

obiekt window

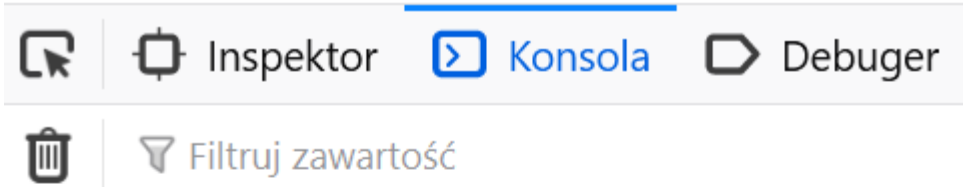


w metodzie setInterval zdefiniowana jest funkcja, która jest wywoływana co 1s

metoda clearInterval kończy ciąg wywołań funkcji zdefiniowanej w setInterval

obiekt navigator

informacje o przeglądarce




```
>> window.navigator.appName;
<- "Netscape"
```

nazwa przeglądarki

```
>> window.navigator.appVersion;
<- "5.0 (Windows)"
```

wersja przeglądarki

```
>> window.navigator.language;
<- "pl"
```

język przeglądarki

```
>> navigator.cookieEnabled;
<- true
```

czy włączone są cookies

```
>> navigator.javaEnabled();
<- false
```

czy włączona jest obsługa Javy

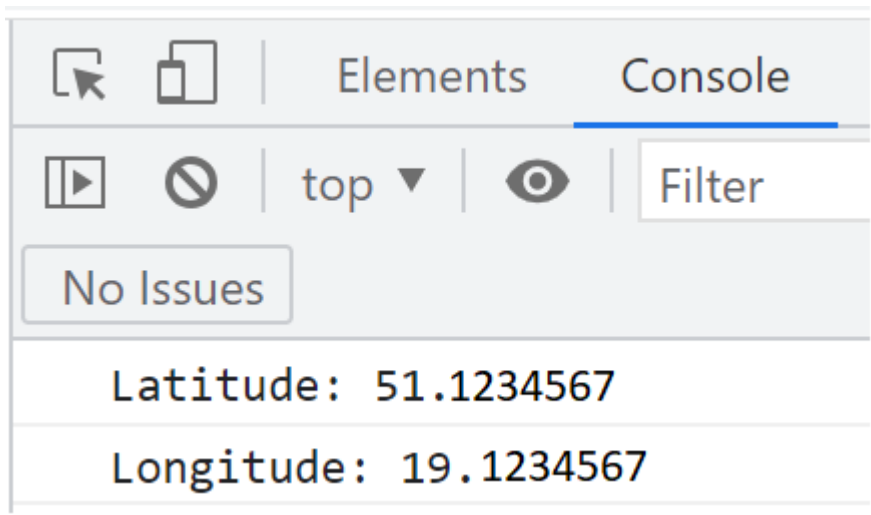
Deprecated. This method always returns false.

https://www.w3schools.com/jsref/obj_navigator.asp

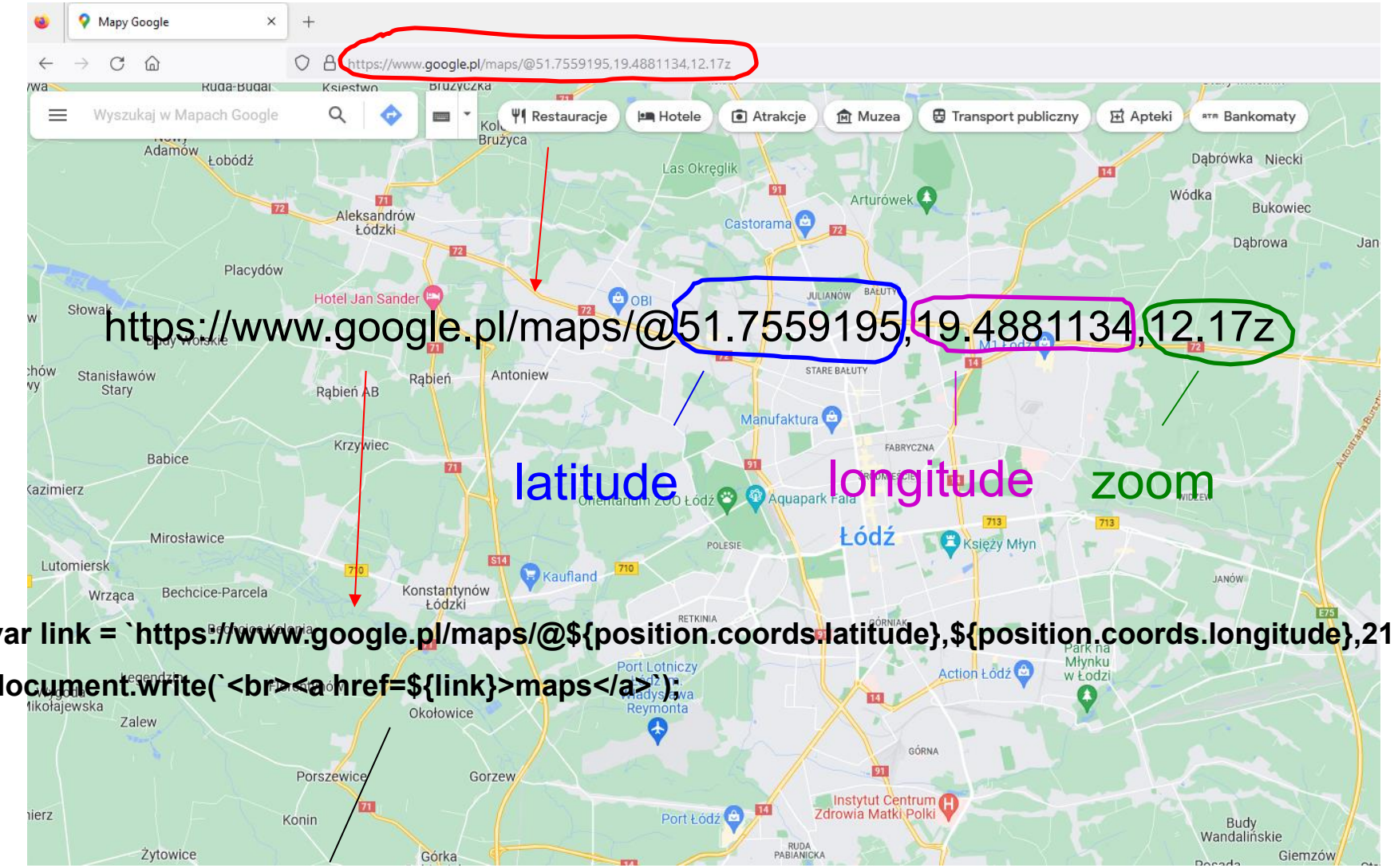
obiekt navigator

```
<!DOCTYPE html>
<html>
<body>
<script>
navigator.geolocation.getCurrentPosition(showPosition);

function showPosition(position) {
  console.log("Latitude: " + position.coords.latitude);
  console.log("Longitude: " + position.coords.longitude);
}
</script>
</body>
</html>
```

długość i szerokość geograficzna pozycji użytkownika



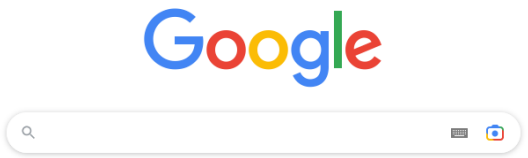
obiekt navigator



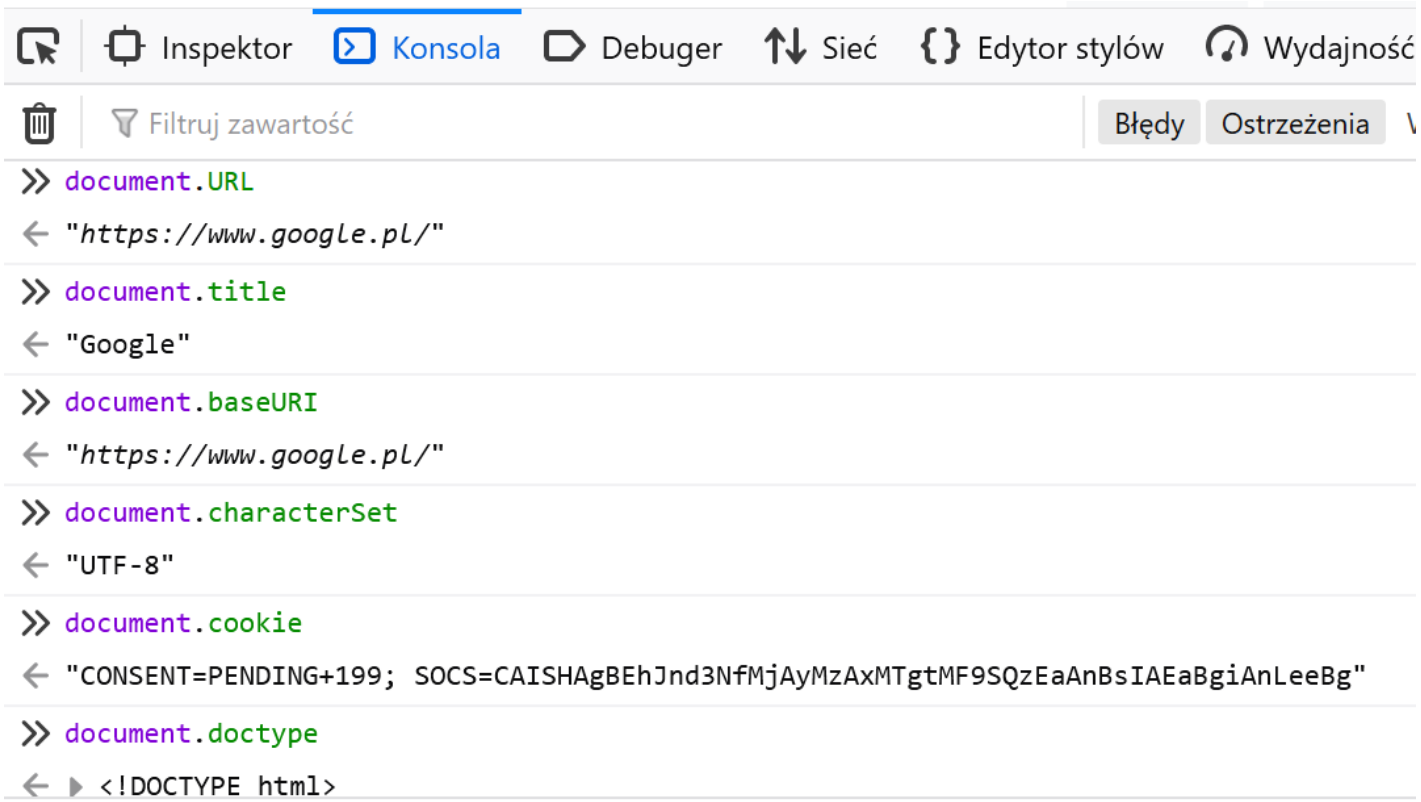

```
var link = `https://www.google.pl/maps/@${position.coords.latitude},${position.coords.longitude},21z`;
document.write(`<br><a href=${link}>maps</a>`);
```

link do mapy google

obiekt document

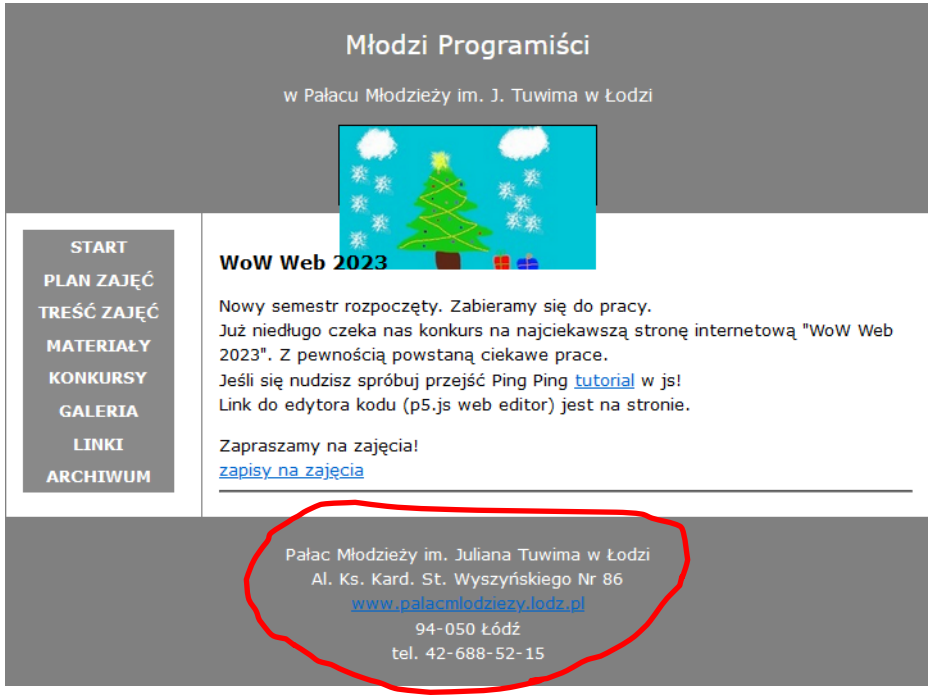


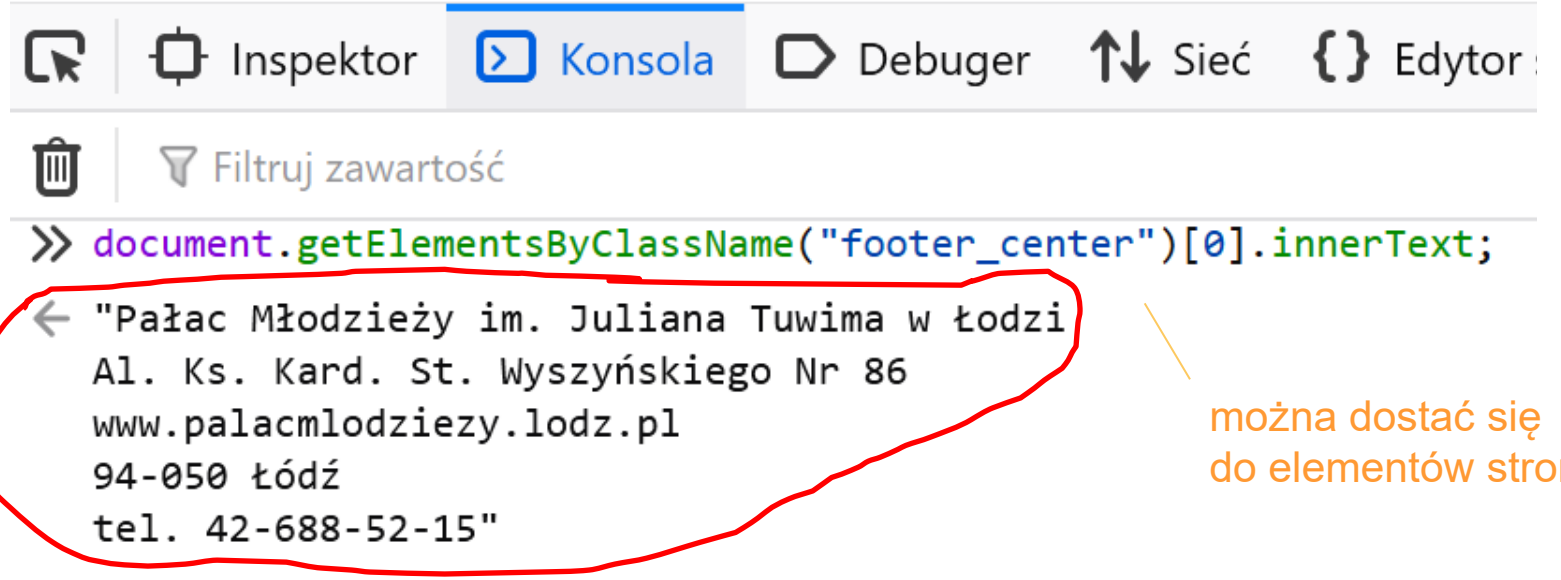
reprezentuje załadowaną do przeglądarki stronę



https://www.w3schools.com/jsref/dom_obj_document.asp

obiekt document





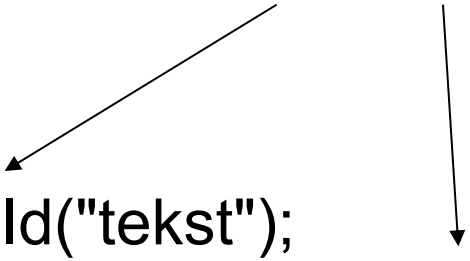
można dostać się
do elementów strony

Dostęp do obiektów strony

```
<p id="tekst">Informatyka jest OK</p>
```

getElementById.html

JavaScript is Case Sensitive !

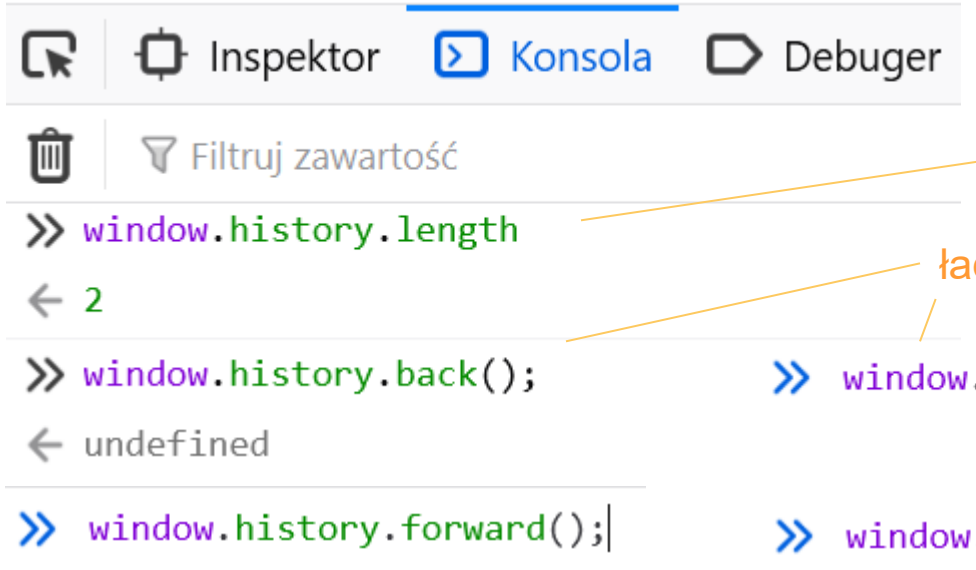


```
<script>
        t=document.getElementById("tekst");
        document.write("Tekst w akapicie " + t.innerHTML);
</script>
```

The easiest way to get or modify the content of an element is by using the innerHTML property.
innerHTML is not a part of the W3C DOM specification. However, it is supported by all major browsers.
The innerHTML property is useful for returning or replacing the content of HTML elements (including <html> and <body>), it can also be used to view the source of a page that has been dynamically modified.

obiekt history

reprezentuje historię odwiedzanych stron w przeglądarce



długość listy odwiedzionych stron

ładuje do przeglądarki poprzedni adres

```
window.history.go(-1);
```

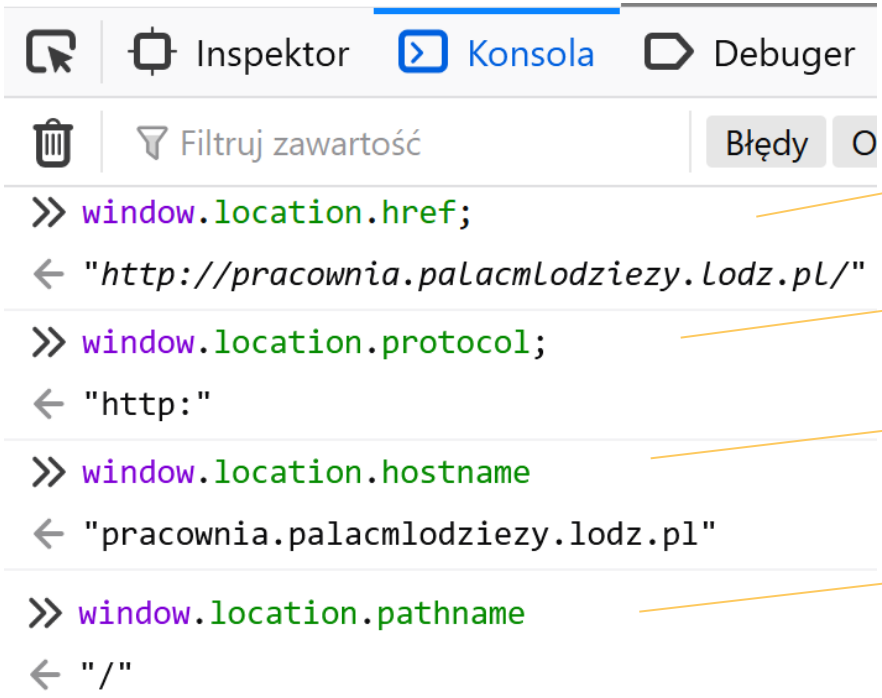
```
window.history.go(1);
```

ładuje do przeglądarki kolejny adres (jeśli istnieje)

https://www.w3schools.com/jsref/api_history.asp

obiekt location

obiektu można użyć do uzyskania aktualnego adresu strony (URL) i przekierowania przeglądarki na nową stronę.



ustawia albo zwraca adres URL

ustawia albo zwraca protokół strony

ustawia albo zwraca nazwa hosta

ustawia albo zwraca ścieżkę

obiekt location

ładuje nową stronę do przeglądarki

```
» location.assign("https://www.w3schools.com");
```

```
» location.replace("https://www.w3schools.com");
```

Różnica między assign() oraz replace():

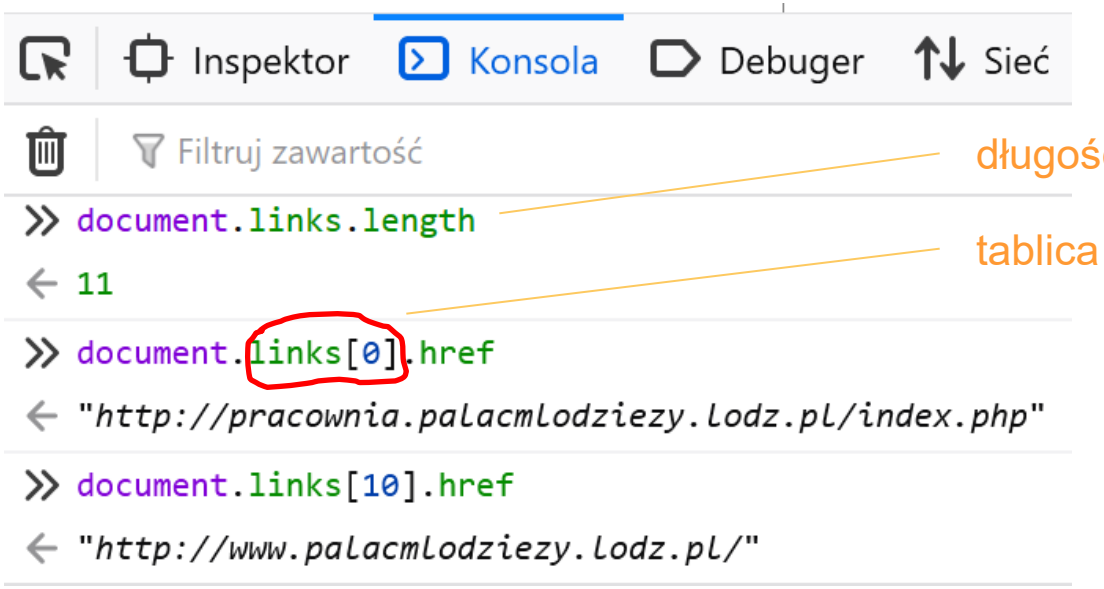
replace() usuwa bieżący adres URL z historii dokumentu.

```
» location.reload();
```

przeładowuje bieżącą stronę

obiekt link

reprezentuje link

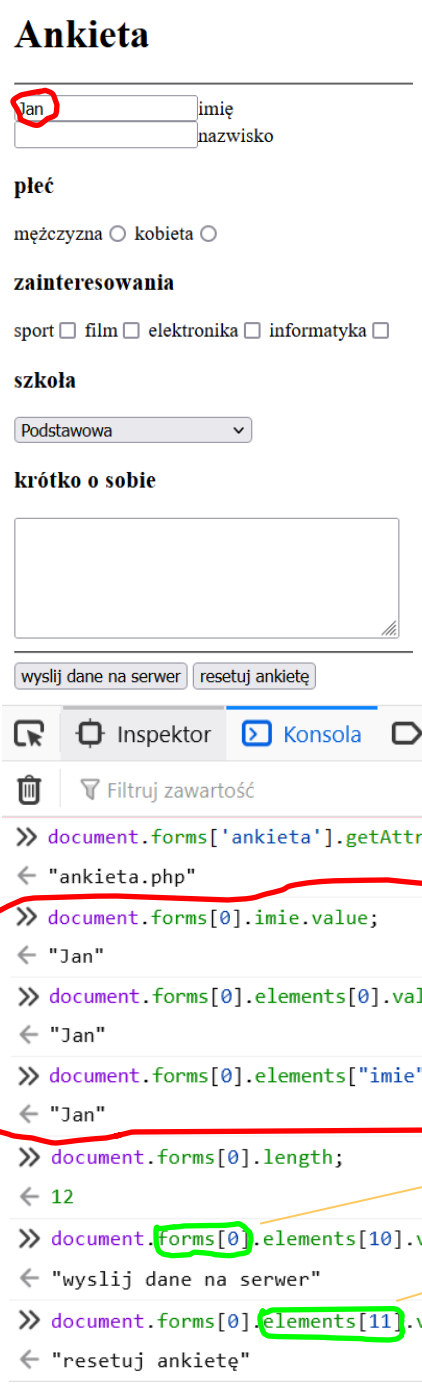


długość tablicy z linkami

tablica z linkami

https://www.w3schools.com/jsref/coll_doc_links.asp

obiekt form



tablica z formularzami

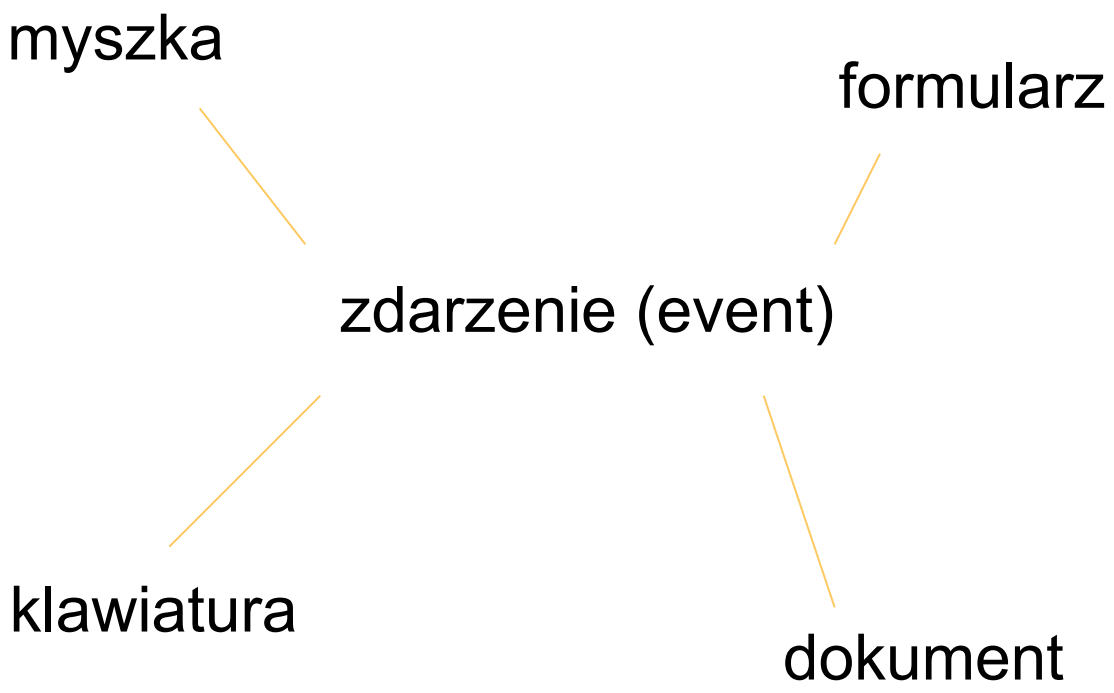
tablica z elementami formularza

reprezentuje formularz

```html
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Ankieta</title>
</head>
<body>
<h1>Ankieta</h1>
<hr>
<form name="ankieta" action="ankieta.php" method="post">
    <input type="text" name="imie">imię<br>
    <input type="text" name="nazwisko">nazwisko<br>

    <h3>płeć</h3>
    mężczyzna<input type="radio" name="plec" value="mężczyzna">
    kobieta<input type="radio" name="plec" value="kobieta">

    <h3>zainteresowania</h3>

    sport<input type="checkbox" name="sport" value="1">
    film<input type="checkbox" name="film" value="1">
    elektronika<input type="checkbox" name="elektronika" value="1">
    informatyka<input type="checkbox" name="informatyka" value="1">

    <h3>szkoła</h3>
    <select name="szkola">
    <option value="podstawowa">Podstawowa</option>
    <option value="liceum">Liceum Poltechniki Łódzkiej</option>
    <option value="politechnika">Politechnika Łódzka</option>
    <option value="uniwersytet">Uniwersytet Łódzki</option>
    <option value="inna">inna</option>
    </select>

    <h3>krótko o sobie</h3>
    <textarea name="o_sobie" rows=5 cols=40></textarea>

    <hr>
    <input type="submit" value="wyslij dane na serwer">
    <input  type="reset" value="resetuj ankietę">
</form>


</body>
</html>
```

https://www.w3schools.com/jsref/dom_obj_form.asp

Zdarzenia



https://www.w3schools.com/jsref/dom_obj_event.asp

Zdarzenia myszy

Zdarzenie	Zdarzenie zachodzi kiedy...
onclick	Użytkownik kliknął na element (np. przycisk)
oncontextmenu	Użytkownik kliknął prawym przyciskiem myszy na element
ondblclick	Użytkownik dwukrotnie kliknął na element
onmousedown	Przycisk myszy jest wciśnięty nad elementem
onmouseenter	Wskaźnik myszy jest przesuwany na element
onmouseleave	Wskaźnik myszy jest przesuwany poza element
onmousemove	Wskaźnik myszy przesuwa się nad elementem
onmouseout	Wskaźnik myszy przesuwa się poza element
onmouseover	Wskaźnik myszy jest przesuwany nad elementem, zdarzenie występuje, gdy wskaźnik myszy wejdzie w element.
onmouseup	Przycisk myszy zostaje zwolniony nad elementem

https://www.w3schools.com/jsref/event_onclick.asp

Zdarzenia klawiatury

onkeydown	Użytkownik naciska klawisz
onkeypress	Użytkownik naciska klawisz
onkeyup	Użytkownik zwalnia klawisz

Warning

The **onkeypress** event is **deprecated**.
It is not fired for all keys (like ALT, CTRL, SHIFT, ESC) in all browsers.
To detect if the user presses a key, always use the **onkeydown** event. It works for all keys.

https://www.w3schools.com/jsref/event_onkeydown.asp

Zdarzenia formularza

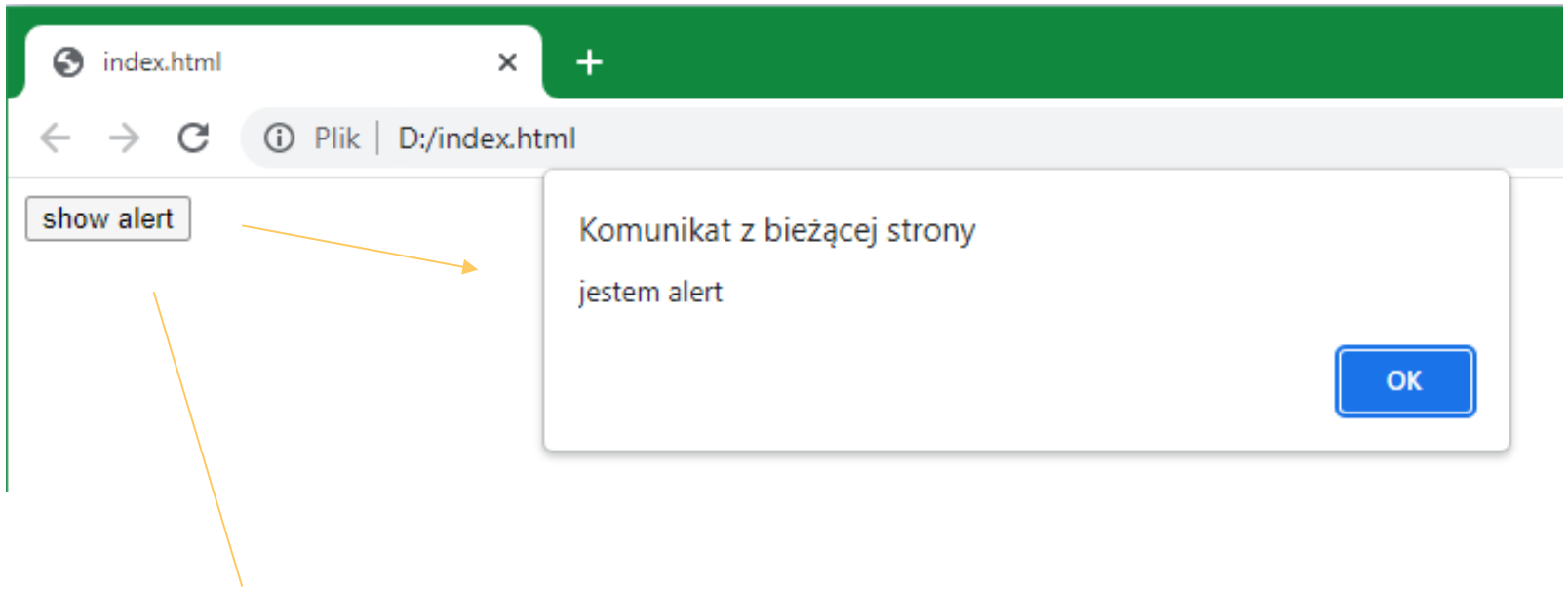
Zdarzenie	Zdarzenie zachodzi kiedy...
onblur	element traci focus
onchange	następuje zmiana wartości elementu
oncontextmenu	użytkownik klika prawym przyciskiem myszy element, aby otworzyć menu kontekstowe
onfocus	element uzyskuje focus
oninput	element pobiera dane wejściowe użytkownika
oninvalid	element jest nieprawidłowy
onreset	po kliknięciu na przycisk reset w formularzu
onsearch	użytkownik wpisze coś w polu wyszukiwania (<input="search">)
onselect	użytkownik zaznaczy tekst w elemencie
onsubmit	formularz zostaje przesłany

https://www.w3schools.com/tags/ref_eventattributes.asp

Zdarzenia dokumentu

Zdarzenie	Zdarzenie zachodzi kiedy...
onload	strona została załadowana
onunload	strona jest zamykana

Zdarzenie onclick



po kliknięciu na przycisk wyświetla się alert z komunikatem

Zdarzenie onclick

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function showAlert(){
            alert("jestem alert");
        }
    </script>
</head>
<body>
    <button onclick="showAlert()">show alert</button>
</body>
</html>
```

po kliknięciu na przycisk wywoływana jest funkcja showAlert()

Zdarzenie onclick

skrypt został umieszczony pod koniec sekcji body – wykona się po wczytaniu całej strony

```
<!DOCTYPE html>
<html>
<body>
    <button id="btn">show alert</button>
    <script>
        let myButton = document.getElementById("btn");
        myButton.onclick = function(){
            alert("jestem alert");
        }
    </script>
</body>
</html>
```

można obsłużyć kliknięcie na przycisk za pomocą właściwości onclick obiektu myButton, który stworzyliśmy za pomocą metody getElementById()

Zdarzenie onclick

```
<!DOCTYPE html>
<html>
<head>

<head>
<body>
    <button id="btn">show alert</button>
    <script>
        let myButton = document.getElementById("btn");
        myButton.addEventListener("click", function(){
                alert("jestem alert");
        });
    </script>
</body>
</html>
```

nazwa zdarzenia

funkcja obsługująca zdarzenie

można obsłużyć kliknięcie na przycisk za pomocą metody addEventListener(), która przyjmuje dwa argumenty: nazwę zdarzenia oraz funkcję je obsługującą

Zdarzenie onclick

zdarzenie DOMContentLoaded jest generowane po załadowaniu strony , czyli w kodzie zdarzenia są widoczne wszystkie elementy strony – również nasz button

```
<!DOCTYPE html>
<html>
<head>
    <script>
        document.addEventListener("DOMContentLoaded", function(){
            let myButton = document.getElementById("btn");
            myButton.onclick = function(){
                alert("jestem alert");
            }
        });
    </script>
</head>
<body>
    <button id="btn">show alert</button>
</body>
</html>
```

Zdarzenie onclick

skrypt obsługujący zdarzenie został umieszczony w oddzielnym pliku main.js

```
<!DOCTYPE html>
<html>
<head>
    <script src="main.js"></script>
</head>
<body>
    <button onclick="showAlert()">show alert</button>
</body>
</html>
```

treść pliku main.js

```
function showAlert(){
    alert("jestem alert");
}
```

Zdarzenie onclick

słowo defer powoduje wykonanie skryptu po wczytaniu strony – elementy strony są wtedy widoczne w skrypcie

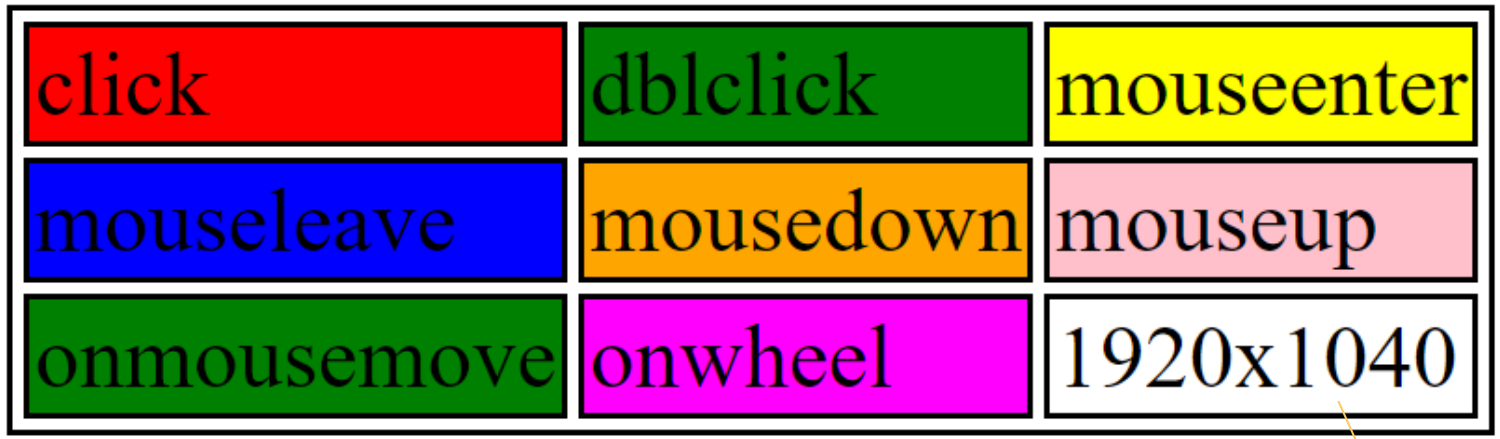
```
<!DOCTYPE html>
<html>
<head>
    <script src="main.js" defer></script>
<head>
<body>
    <button id="btn">show alert</button>
</body>
</html>
```

treść pliku main.js

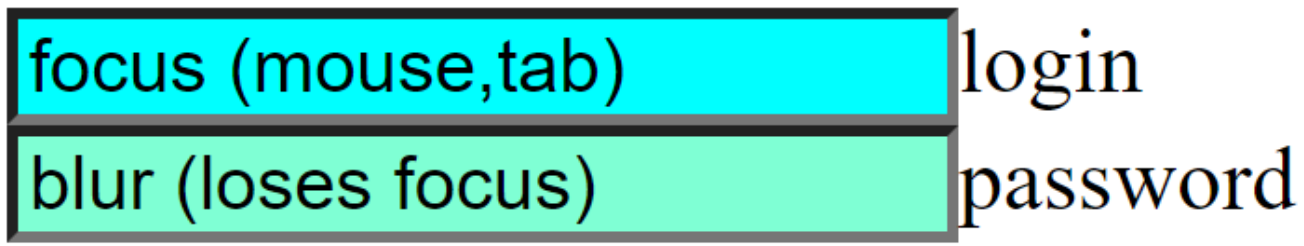
```
let myButton = document.getElementById("btn");
myButton.onclick = function(){
    alert("jestem alert");
}
```

Zdarzenia

napisz stronę z obsługą kilku zdarzeń



wymiary okna



login

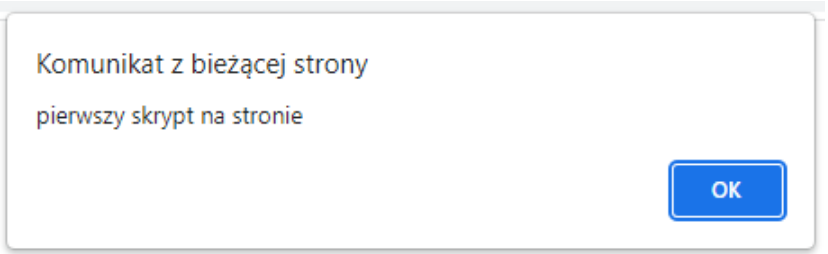
password

Kolejność wykonywania skryptów

```
alert("pierwszy skrypt na stronie")
```

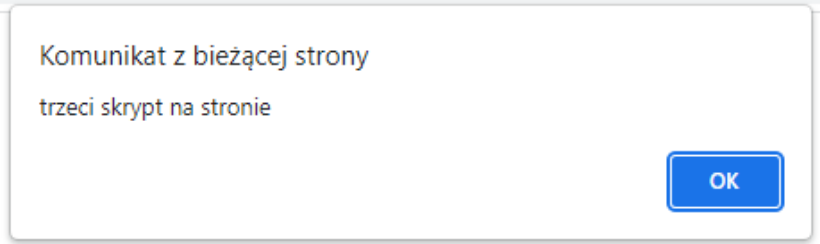
```
<!DOCTYPE html>
<html>
<head>
    <script src="main.js"></script>
<head>
<body>
    <script> alert("drugi skrypt na stronie");</script>
    <h1>Moja strona</h1>
    <script>
        alert("trzeci skrypt na stronie");
    </script>
</body>
</html>
```

1

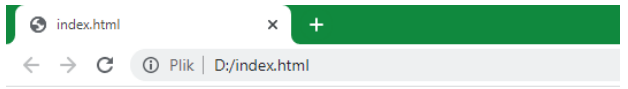


2

3



skrypty w sekcji <head> wykonują się jako pierwsze, potem kolejno następne według kolejności.

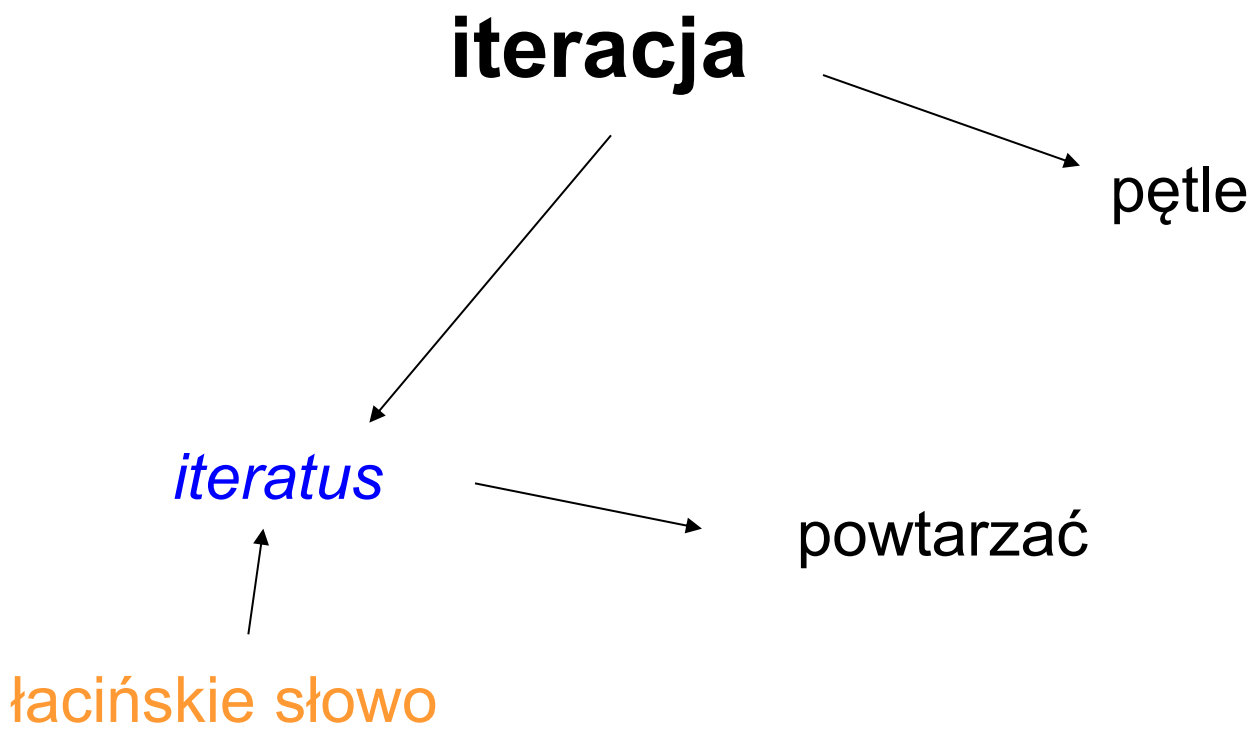


Moja strona

REKURENCJA

ITERACJA

Iteracja



Zadanie

Wartość silni określa wzór znany z lekcji matematyki:

$$n! = 1 * 2 * 3 * 4 * (n - 1) * n$$

Napisz funkcję obliczającą silnię zadanej liczby metodą iteracyjną.

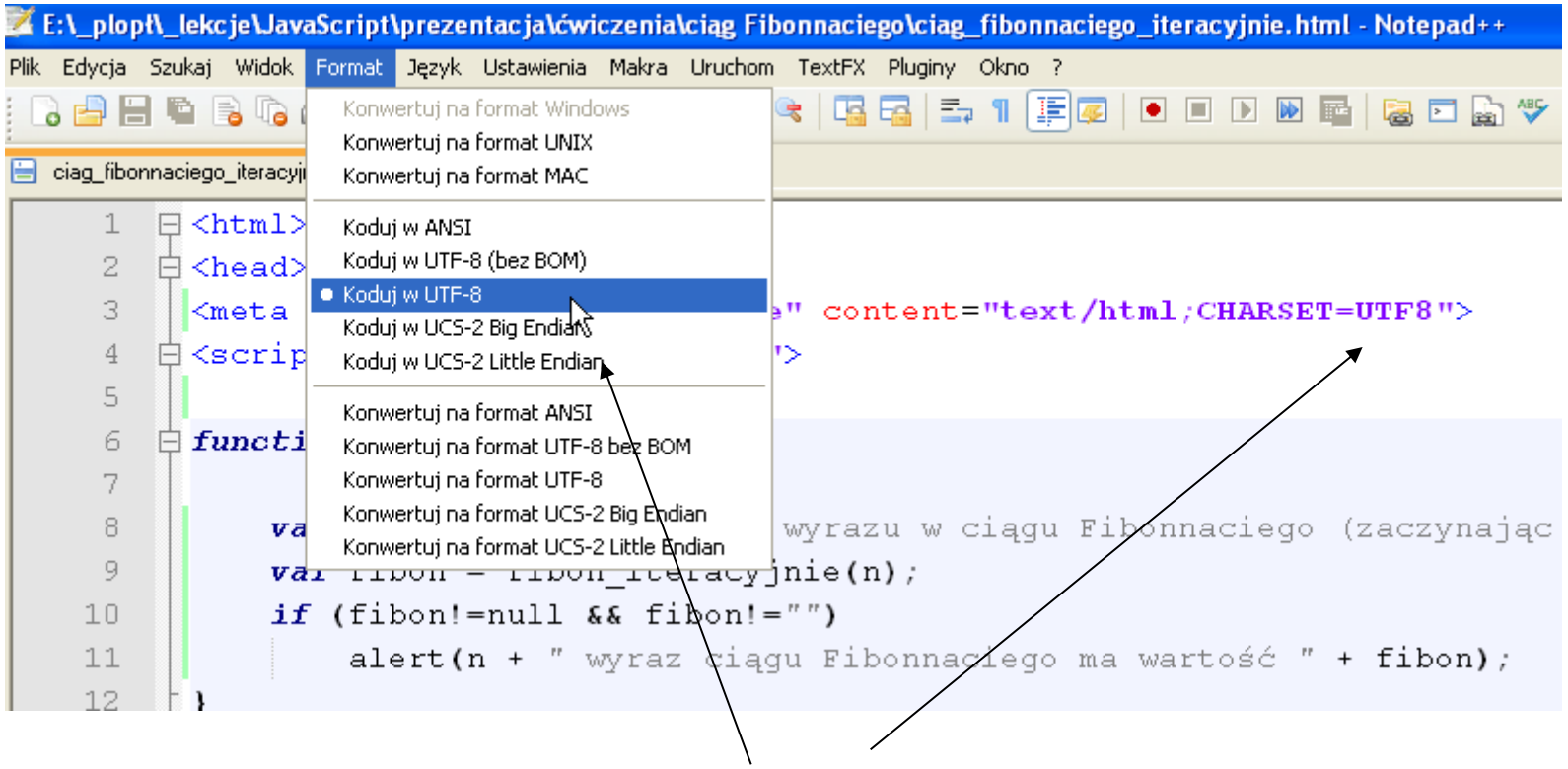
Silnia iteracyjnie

$$n! = 1 * 2 * 3 * 4 * (n - 1) * n$$

funkcja obliczająca iteracyjnie wartość silni

```
function silnia_iteracyjnie (n){

        var silnia=1;
        for (i=1; i<=n; i++)
                silnia=silnia * i;
        return silnia;

}
```

Polskie znaki w Notepad++



gwarancja polskich znaków na stronie www

Silnia iteracyjnie

silnia_iteracyjnie.html

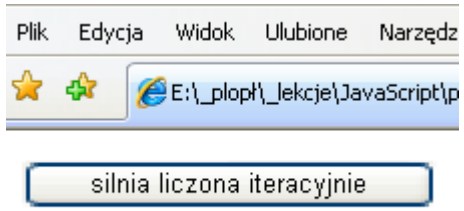
```
1  <html>
2  <head>
3  <meta HTTP-EQUIV="Content-Type" content="text/html;CHARSET=UTF8">
4
5  <script type="text/javascript">
6
7
8  function licz(){
9
10     var n=prompt("podaj liczbę całkowitą większą od zera","");
11     var silnia = silnia_iteracyjnie(n);
12     if (silnia!=null && silnia!="")
13         alert("silnia liczby " + n + " wynosi " + silnia);
14 }
15
16 function silnia_iteracyjnie(n){
17
18
19     var silnia=1;
20     for (i=1; i<=n; i++)
21         silnia=silnia * i;
22     return silnia;
23
24 }
25
26 </script>
27 </head>
28 <body>
29 <input type="button" onclick="licz()" value="silnia liczona iteracyjnie"><br />
30 </body>
31 </html>
```

wywołanie funkcji silnia_iteracyjnie(n)

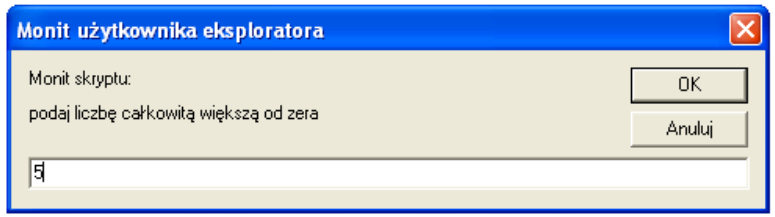
po naciśnięciu przycisku wywoływana jest funkcja licz()

przycisk

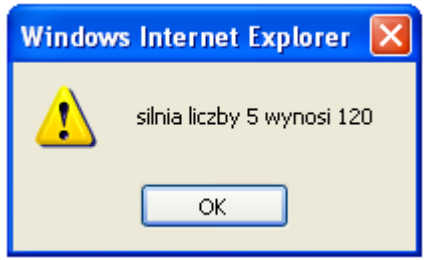
Silnia iteracyjnie



przycisk

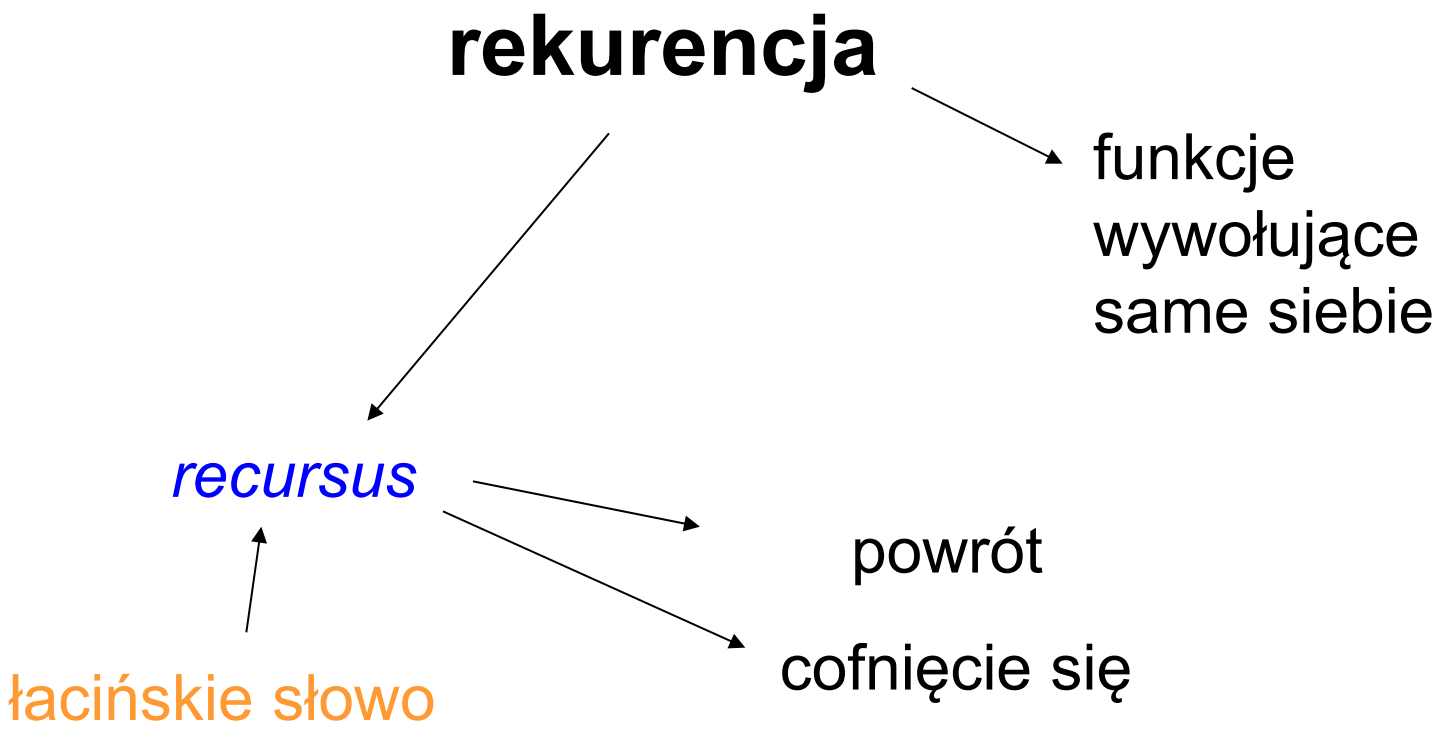


okienko prompt



okienko alert

Rekurencja


rekurencja
funkcje
wywołujące
same siebie
recursus
powrót
cofnięcie się
łacińskie słowo

Zadanie

Napisz funkcję obliczającą silnię zadanej liczby za pomocą rekurencji.

Silnia rekurencyjnie

$3! = 1 * 2 * 3$

$4! = 1 * 2 * 3 * 4$

3!

$4! = 3! * 4$

$n! = (n-1)! * n$

silnia_rekurencyjnie (n)

silnia_rekurencyjnie (n - 1)

Silnia rekurencyjnie

n należy do liczb naturalnych

$$\text{silnia\_rekurencyjnie}(n) = \begin{cases} 1, & \text{gdy } n=1, \\ \text{silnia\_rekurencyjnie}(n-1) * n, & \text{gdy } n>1 \end{cases}$$

Uwaga!
Przyjąłem, że zero nie należy do liczb naturalnych
W matematyce przyjmuje się 0!=1

Silnia rekurencyjnie

funkcja obliczająca rekurencyjnie wartość silni

```
function silnia_rekurencyjnie (n){

        if (n==1)
            return 1;
          else
            return silnia_rekurencyjnie(n-1) * n

}
```

Silnia rekurencyjnie

silnia_rekurencyjnie.html

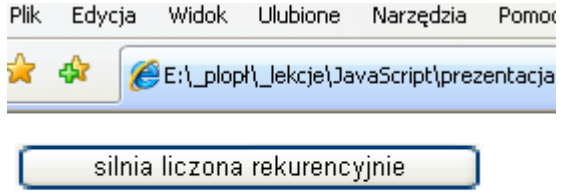
```
1  <html>
2  <head>
3  <meta HTTP-EQUIV="Content-Type" content="text/html;CHARSET=UTF8">
4
5  <script type="text/javascript">
6
7
8  function licz(){
9
10     var n=prompt("podaj liczbę całkowitą większą od zera","");
11     var silnia = silnia_rekurencyjnie(n);
12     if (silnia!=null && silnia!="")
13         alert("silnia liczby " + n + " wynosi " + silnia);
14 }
15
16 function silnia_rekurencyjnie (n){
17
18     if (n==1)
19         return 1;
20       else
21         return silnia_rekurencyjnie(n-1) * n
22
23 }
24
25
26 </script>
27 </head>
28 <body>
29 <input type="button" onclick="licz()" value="silnia liczona rekurencyjnie"><br />
30 </body>
31 </html>
```

wywołanie funkcji silnia_rekurencyjnie(n)

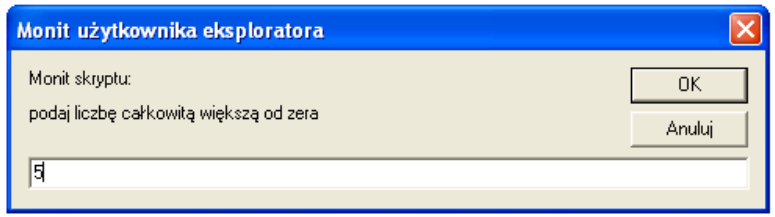
przycisk

po naciśnięciu przycisku wywoływana jest funkcja licz()

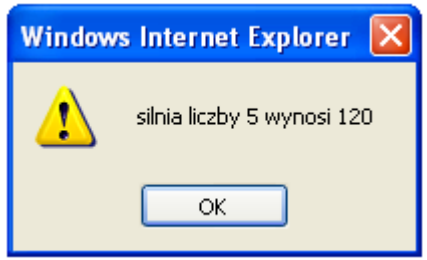
Silnia rekurencyjnie



przycisk



okienko prompt



okienko alert

Ciąg Fibonacciego

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987…

niektórzy matematycy
zaczynają od zera

Czy możesz napisać wzór na ciąg?

Ciąg Fibonacciego

numer wyrazu w ciągu

↓

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987…

pierwszy wyraz ciągu : 1

drugi wyraz ciągu : 1

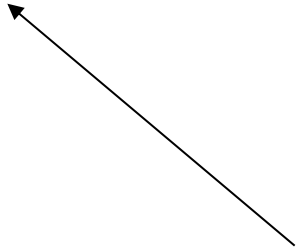
następny wyraz ciągu : suma dwóch poprzednich

Zadanie

Napisz funkcję obliczającą zadany wyraz ciągu Fibonacciego iteracyjnie.

Ciąg Fibonacciego iteracyjnie

```
function fibon_iteracyjnie (n){

        if (n <= 2)
            return 1;
          else {
            f1=1;
            f2=1;
            for (i=3; i<=n; i++){
                f=f1 + f2;
                f1=f2;
                f2=f;
            }
            return f;
         }
}
```



funkcja obliczająca iteracyjnie wartość n-tego wyrazu ciągu

Ciąg Fibonacciego iteracyjnie

ciag_fibonnaciego_iteracyjnie.html

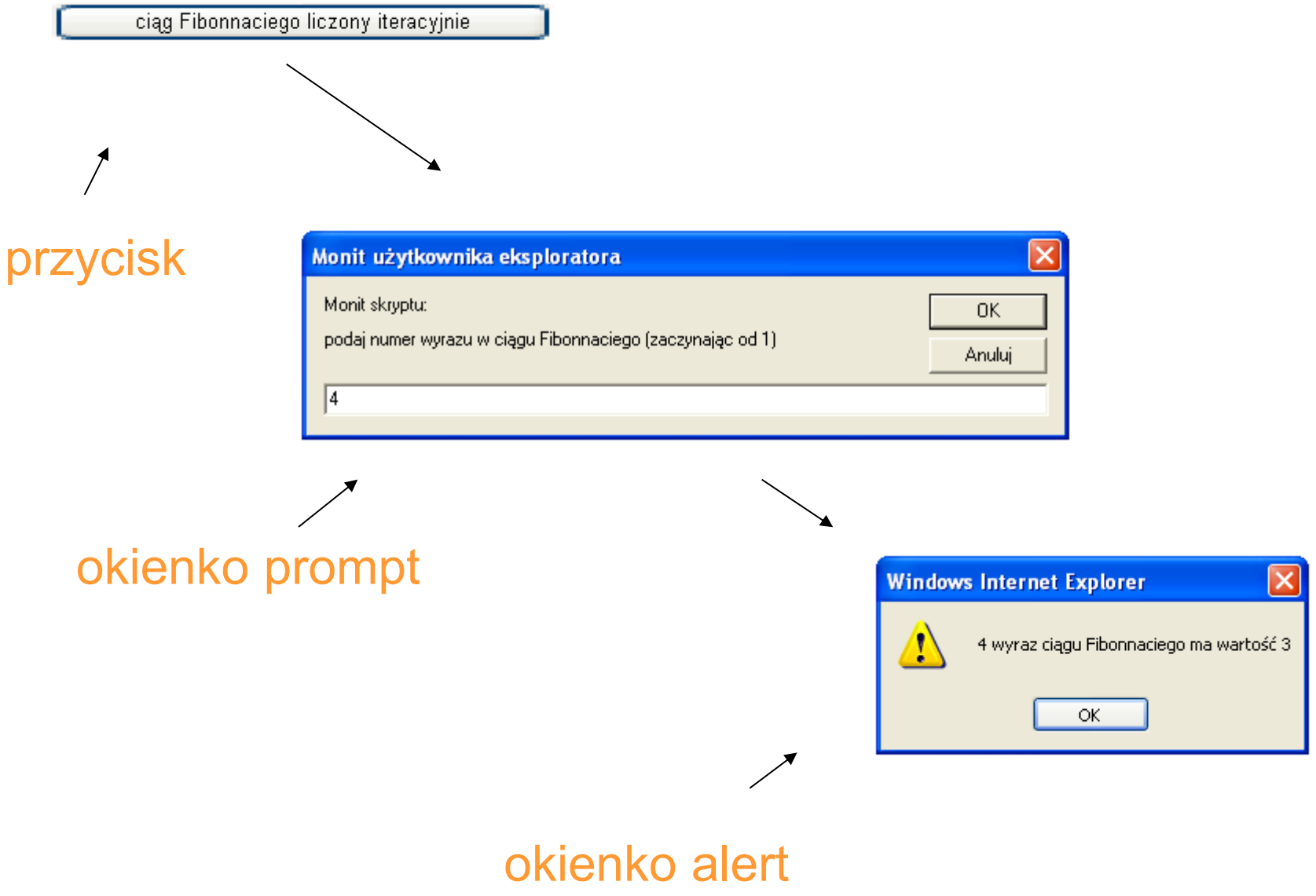
```
1  <html>
2  <head>
3  <meta HTTP-EQUIV="Content-Type" content="text/html;CHARSET=UTF8">
4  <script type="text/javascript">
5
6  function licz(){
7
8      var n=prompt("podaj numer wyrazu w ciągu Fibonnaciego (zaczynając od 1)","");
9      var fibon = fibon_iteracyjnie(n);
10     if (fibon!=null && fibon!="")
11         alert(n + " wyraz ciągu Fibonnaciego ma wartość " + fibon);
12 }
13
14 function fibon_iteracyjnie (n){
15
16     if (n <= 2)
17         return 1;
18         else {
19             f1=1;
20             f2=1;
21             for (i=3; i<=n; i++){
22                 f=f1 + f2;
23                 f1=f2;
24                 f2=f;
25
26             }
27             return f;
28         }
29 }
30
31
32 </script>
33 </head>
34 <body>
35 <input type="button" onclick="licz()" value="ciąg Fibonnaciego liczony iteracyjnie"><br />
36 </body>
37 </html>
```

wywołanie funkcji fibon_iteracyjnie(n)

przycisk

po naciśnięciu przycisku wywoływana jest funkcja licz()

Ciąg Fibonacciego iteracyjnie

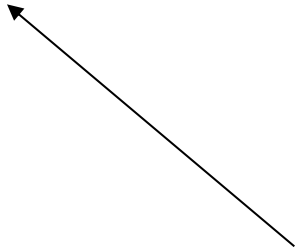


Zadanie

Napisz funkcję obliczającą zadany wyraz ciągu Fibonacciego za pomocą rekurencji.

Ciąg Fibonacciego rekurencyjnie

```
function fibon_rekurencyjnie(n){

    if (n<=2)
      return 1;
     else
      return fibon_rekurencyjnie(n-2) + fibon_rekurencyjnie(n-1)

}
```



funkcja obliczająca rekurencyjnie wartość n-tego wyrazu ciągu

Ciąg Fibonacciego rekurencyjnie

ciag_fibonnaciego_rekurencyjnie.html

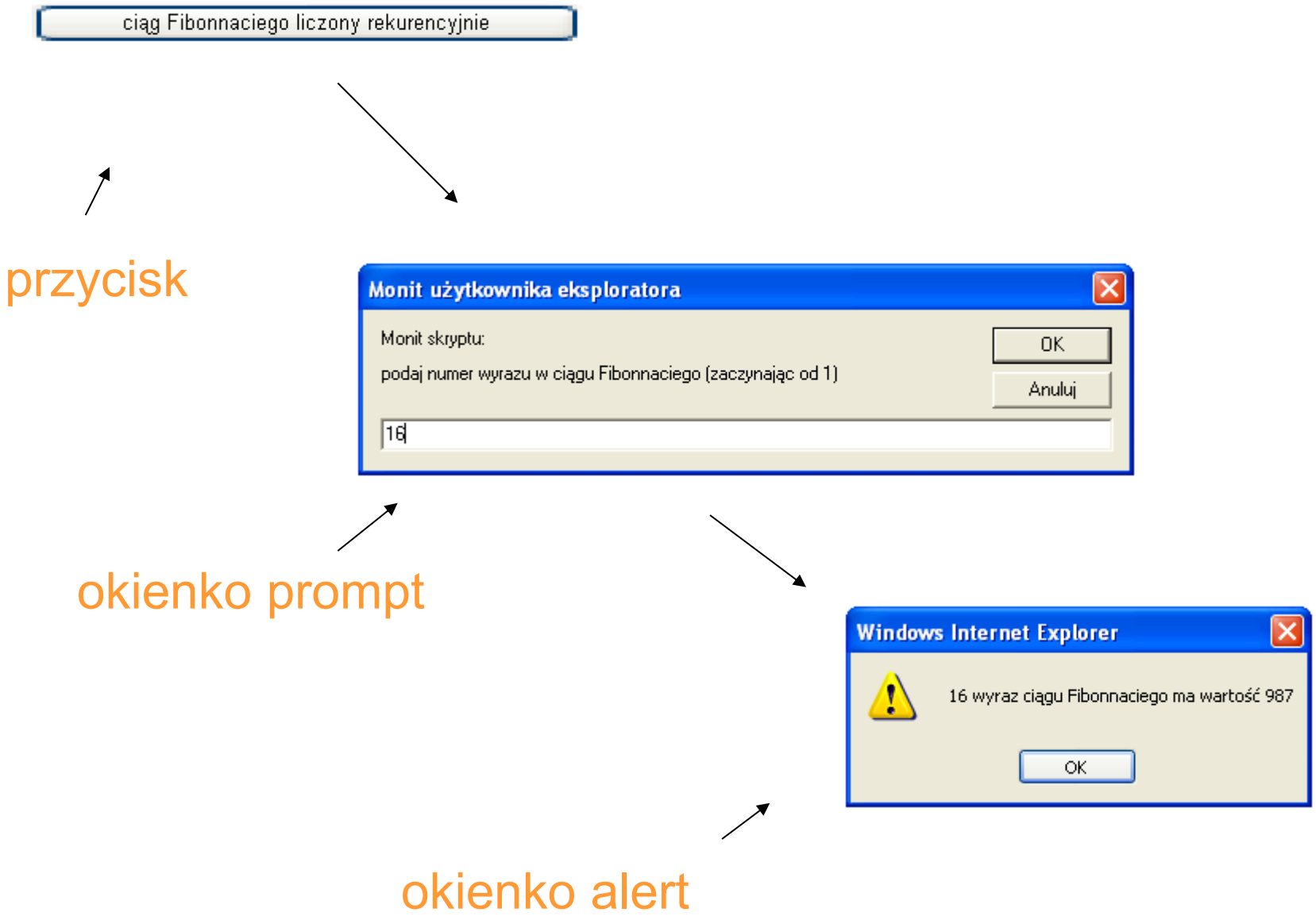
```
1  <html>
2  <head>
3  <meta HTTP-EQUIV="Content-Type" content="text/html;CHARSET=UTF8">
4  <script type="text/javascript">
5
6  function licz(){
7
8      var n=prompt("podaj numer wyrazu w ciągu Fibonnaciego (zaczynając od 1)","");
9      var fibon = fibon_rekurencyjnie(n);
10     if (fibon!=null && fibon!="")
11         alert(n + " wyraz ciągu Fibonnaciego ma wartość " + fibon);
12 }
13
14 function fibon_rekurencyjnie(n){
15
16     if (n<=2)
17        return 1;
18     else
19        return fibon_rekurencyjnie(n-2) + fibon_rekurencyjnie(n-1)
20
21 }
22
23
24 </script>
25 </head>
26 <body>
27 <input type="button" onclick="licz()" value="ciąg Fibonnaciego liczony rekurencyjnie"><br />
28 </body>
29 </html>
```

wywołanie funkcji fibon_rekurencyjnie(n)

przycisk

po naciśnięciu przycisku wywoływana jest funkcja licz()

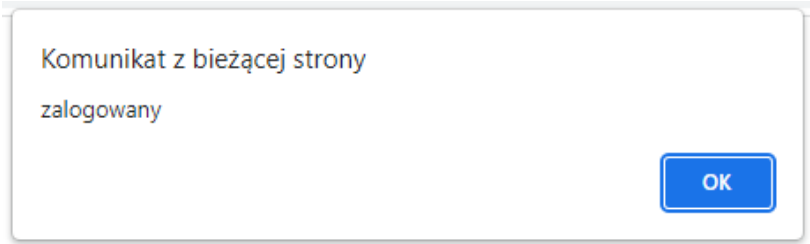
Ciąg Fibonacciego rekurencyjnie

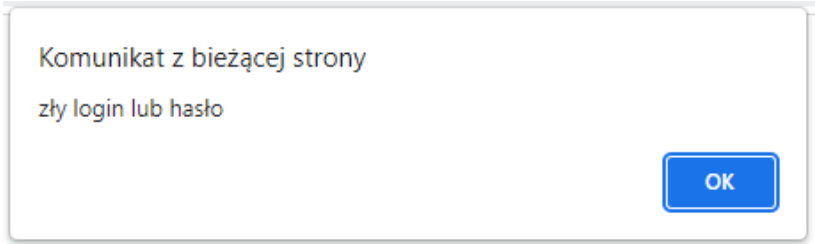


logowanie

dołóż kolejny stopień uwierzytelnienia użytkownika (np. kolejne pole input)

formularz logowania





strona z informacją pojawia się po zalogowaniu

tajna informacja

logowanie

logowanie.html

```html
<!doctype html>
<html>
<head>
    <title>pole textowe</title>
    <meta charset="utf-8">
    <script>
        function wyswietl(){
            let login = document.forms["zaloguj"]["login"].value;
            let password = document.forms["zaloguj"]["password"].value;
            //alert("twój login to: " + login);
            //alert("twoje hasło to: " + password);
            if(login == "aaa" && password == "bbb"){
                alert("zalogowany");
                window.location = 'tajne.html'
            }else{
                alert("zły login lub hasło");
            }
        }
    </script>
</head>
<body>
    <form name="zaloguj" method="post">
            login: <input type="text" name="login" placeholder="login" size="30">
            <br><br>
            hasło: <input type="password" name="password" placeholder="password" size="30">
            <br><br>
            <input type="button" value="login" onclick="wyswietl()"/>
    </form>
</body>
```

otworzenie strony tajne.html po zalogowaniu

tajne.html

```html
<!doctype html>
<html>
<head>
    <title>pole textowe</title>
    <meta charset="utf-8">
</head>
<body>
<h1>tajna informacja</h1>
</body>
```

ankieta

na podstawie przedstawionej ankiety stwórz własną na interesujący Cię temat

Ankieta

Małgorzata imię
Kowalska nazwisko

płeć

mężczyzna ○ kobieta ◉

zainteresowania

sport ☑ film ☑ elektronika ☐ informatyka ☑

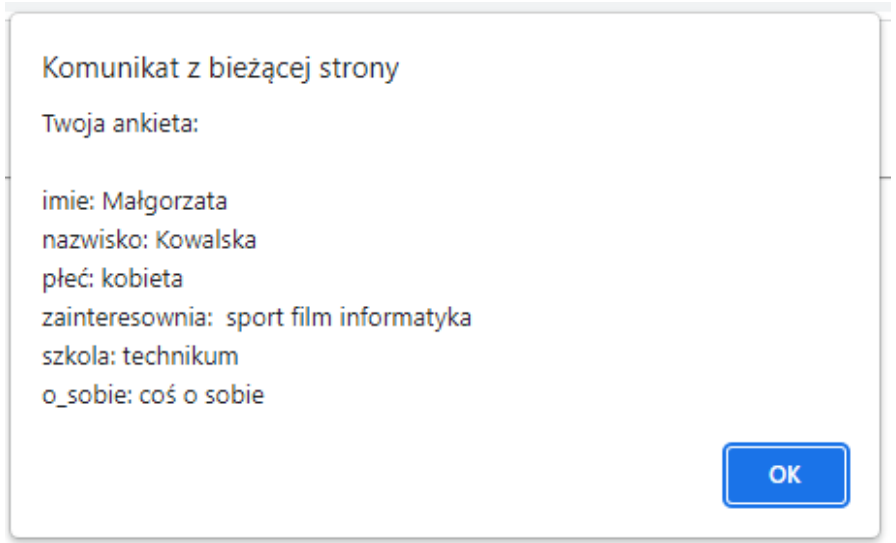
szkoła

Technikum

krótko o sobie

coś o sobie

pokaż ankietę | resetuj ankietę



ankieta

plik
ankieta.html

formularz
ankiety

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Ankieta</title>
    <script src='main.js'></script>
</head>
<body>
<h1>Ankieta</h1>
<hr>
<form name="ankieta" method="post">
    <input type="text" name="imie">imię<br>
    <input type="text" name="nazwisko">nazwisko<br>

    <h3>płeć</h3>
    mężczyzna<input type="radio" name="plec" value="mężczyzna">
    kobieta<input type="radio" name="plec" value="kobieta">

    <h3>zainteresowania</h3>

    sport<input type="checkbox" name="sport" value="1">
    film<input type="checkbox" name="film" value="1">
    elektronika<input type="checkbox" name="elektronika" value="1">
    informatyka<input type="checkbox" name="informatyka" value="1">

    <h3>szkoła</h3>
    <select name="szkola">
    <option value="podstawowa">Podstawowa</option>
    <option value="technikum">Technikum</option>
    <option value="liceum">Liceum</option>
    <option value="politechnika">Politechnika Łódzka</option>
    <option value="uniwersytet">Uniwersytet Łódzki</option>
    <option value="inna">inna</option>
    </select>

    <h3>krótko o sobie</h3>
    <textarea name="o_sobie" rows=5 cols=40></textarea>

    <hr>
    <input type="button" value="pokaż ankietę" onClick="pokaz()"/>
    <input  type="reset" value="resetuj ankietę">
</form>

</body>
</html>
```

ankieta

plik main.js

```
function pokaz(){
    let imie = document.forms["ankieta"]["imie"].value;
    let nazwisko = document.forms["ankieta"]["nazwisko"].value;
    let plec = document.forms["ankieta"]["plec"].value;
    let sport = document.forms["ankieta"]["sport"];
    let film = document.forms["ankieta"]["film"];
    let elektronika = document.forms["ankieta"]["elektronika"];
    let informatyka = document.forms["ankieta"]["informatyka"];
    let szkola = document.forms["ankieta"]["szkola"].value;
    let o_sobie = document.forms["ankieta"]["o_sobie"].value;


    // zbieramy dane z ankiety
    let zainteresowania = "";
    if (sport.checked == true){
      zainteresowania += " sport";
    }
    if (film.checked == true){
      zainteresowania += " film";
    }
    if (elektronika.checked == true){
      zainteresowania += " elektronika";
    }
    if (informatyka.checked == true){
      zainteresowania += " informatyka";
    }
    alert("Twoja ankieta:\n" +
          "\nimie: " + imie +
          "\nnazwisko: " + nazwisko +
          "\npłeć: " + plec +
          "\nzainteresownia: " + zainteresowania +
          "\nszkola: " + szkola +
          "\no_sobie: " + o_sobie
         );

}
```

pobieranie przesłanych z formularza danych do zmiennych

zebranie danych z checkboxów

alert wyświetlający dane

quiz

Quiz

Które oprogramowanie nie jest systemem zarządzania treścią (CMS)?

- ○ Apache
- ○ Joomla
- ○ WordPress
- ○ Mambo

sprawdź

tu pojawi się odpowiedź

na podstawie przedstawionego quizu stwórz własny z kilkoma pytaniami

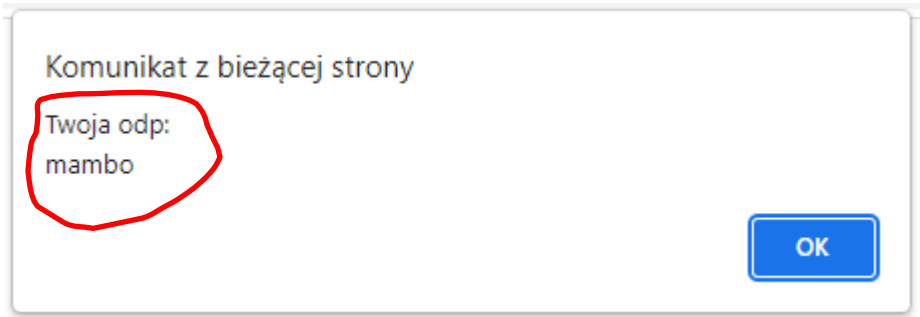
Quiz

Które oprogramowanie nie jest systemem zarządzania treścią (CMS)?

- ○ Apache
- ○ Joomla
- ○ WordPress
- ◉ Mambo

sprawdź

źle



quiz

plik quiz.html

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Quiz</title>
    <script src='main.js'></script>
</head>
<body>
<h1>Quiz</h1>

<form name="quiz" method="post">

    <h3>Które oprogramowanie nie jest systemem zarządzania treścią (CMS)?</h3>
    <input type="radio" name="q1" value="apache">  Apache<br>
    <input type="radio" name="q1" value="joomla">  Joomla<br>
    <input type="radio" name="q1" value="wordpress">  WordPress<br>
    <input type="radio" name="q1" value="mambo">  Mambo<br><br>

    <input type="button" value="sprawdź" onClick="sprawdz()"/>
</form>

<p id="odp">tu pojawi się odpowiedź</p>

</body>
</html>
```

quiz

plik main.js

```
function sprawdz(){
    let q1 = document.forms["quiz"]["q1"].value;
    let odp = document.getElementById("odp");

    if(q1=="apache"){
        odp.innerHTML="dobrze";
    }else{
        odp.innerHTML="źle";
    }

    alert("Twoja odp:\n" + q1);
}
```

calc

kalkulator obsługuje podstawowe działania

formularz select

zabezpieczenie przed dzieleniem przez zero

dołącz do kalkulatora modulo, power, min, max oraz inne funkcje matematyczne z biblioteki Math

calc

plik calc.html

```html
<!DOCTYPE html>
<html>
    <head>
        <title>kalkulator</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="main.css">
        <script src="main.js" defer></script>
    </head>
    <body>
    <h1>kalkulator</h1>
        <input id="a" type="number">
        <select id="op">
            <option>+</option>
            <option>-</option>
            <option>*</option>
            <option>/</option>
        </select>
        <input id="b" type="number">
        <label>=</label>
        <label id="wynik"></label><br>
        <button id="btn">oblicz</button>
    </body>
</html>
```

formularz select

calc

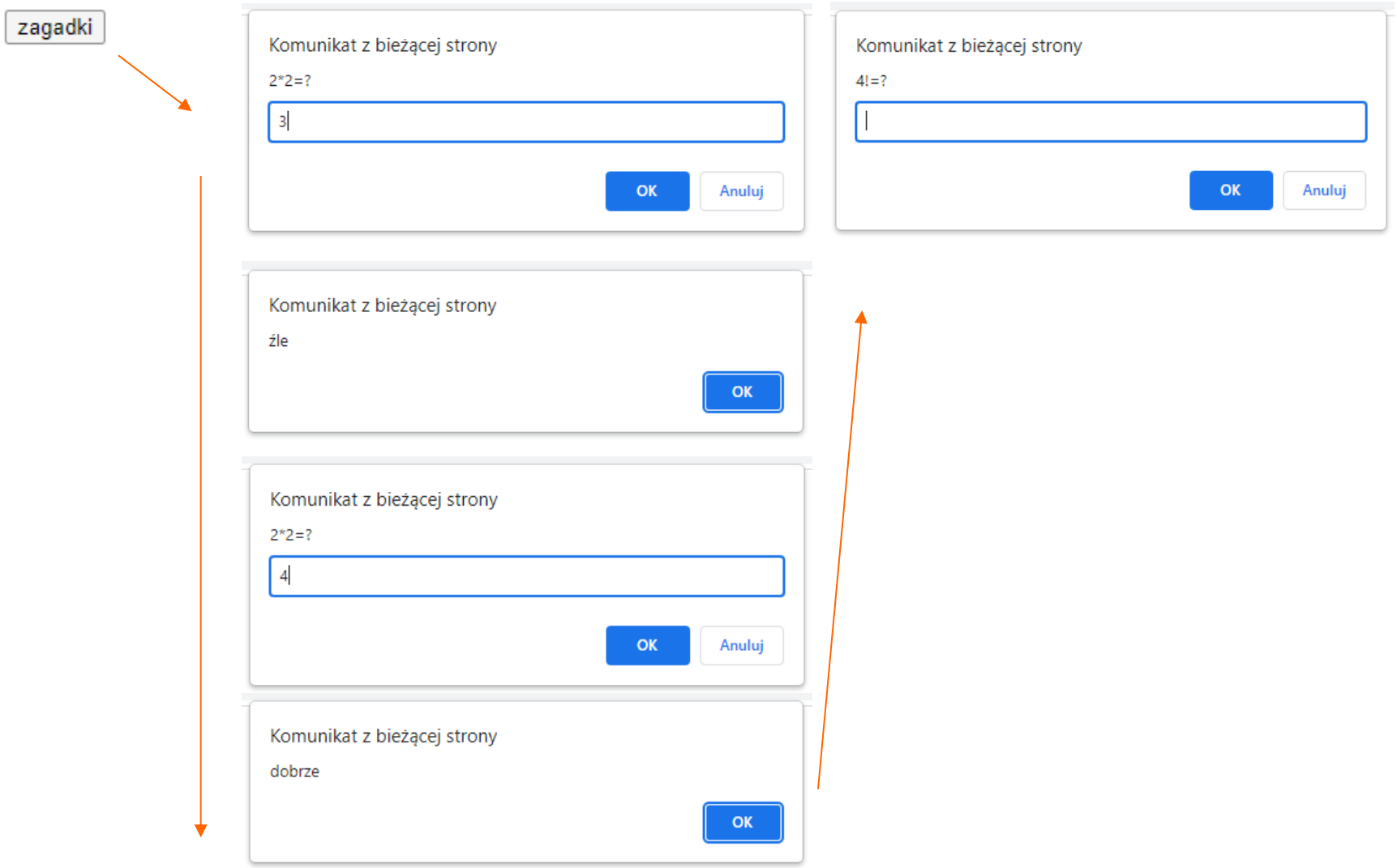
plik main.js

```
let btn = document.getElementById("btn");

btn.onclick = function(){
    let a = document.getElementById("a").value;
    let b = document.getElementById("b").value;
    let op = document.getElementById("op").value;
    switch (op){
        case "+":
            rezultat = parseFloat(a) + parseFloat(b);
            break;
        case "-":
            rezultat = parseFloat(a) - parseFloat(b);
            break;
        case "*":
            rezultat = parseFloat(a) * parseFloat(b);
            break;
        case "/":
            if (b != "0"){
                rezultat = parseFloat(a) / parseFloat(b);
            }else{
                rezultat = "nie dzielimy przez zero!"
            }
            break;
    }
    document.getElementById("wynik").innerHTML = rezultat;
}
```

konwersja z typu string do liczby zmiennoprzecinkowej

zabezpieczenie przed dzieleniem przez zero

zagadki


zagadki
Komunikat z bieżącej strony
2*2=?
3
OK
Anuluj
Komunikat z bieżącej strony
4!=?
OK
Anuluj
Komunikat z bieżącej strony
źle
OK
Komunikat z bieżącej strony
2*2=?
4
OK
Anuluj
Komunikat z bieżącej strony
dobrze
OK

zagadki

plik zagadki.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>Zagadki</title>
    <script src='main.js'></script>
</head>
<body>
    <button onclick="zagadki()">zagadki</button>
</body>
</html>
```

zagadki

plik main.js

```
const zagadki = function(){
    z("2*2=?","4");
    z("4!=?","24");
    z("Bitwa pod Grunwaldem","1410");
    z("Nazwisko autora powieści 'Lalka'", "Prus");
    z("Największa rzeka w Polsce","Wisła");
}

const z = function(zag, odp){
    let isOk = false;
    while(!isOk){
        let odpowiedz = prompt(zag);
        if (odpowiedz == odp){
            alert("dobrze");
            isOk=true;
        }else{
            alert("źle");
        }
    }
}
```

opracuj inny sposób serwowania zagadek użytkownikowi (np. zagadki w tabeli oraz iteracja po niej)

cookies (ciasteczka)

Plik cookie to mały plik, który serwer osadza na komputerze użytkownika. Za każdym razem, gdy przeglądarka zażąda strony internetowej z serwera, do żądania dodawane są pliki cookie należące do strony. W ten sposób serwer może zapisać informacje na komputerze użytkownika i mieć do nich dostęp.

Plik cookie jest często używany do identyfikacji użytkownika.

Pliki cookie są zapisywane w formie par nazwa=wartość, np.:

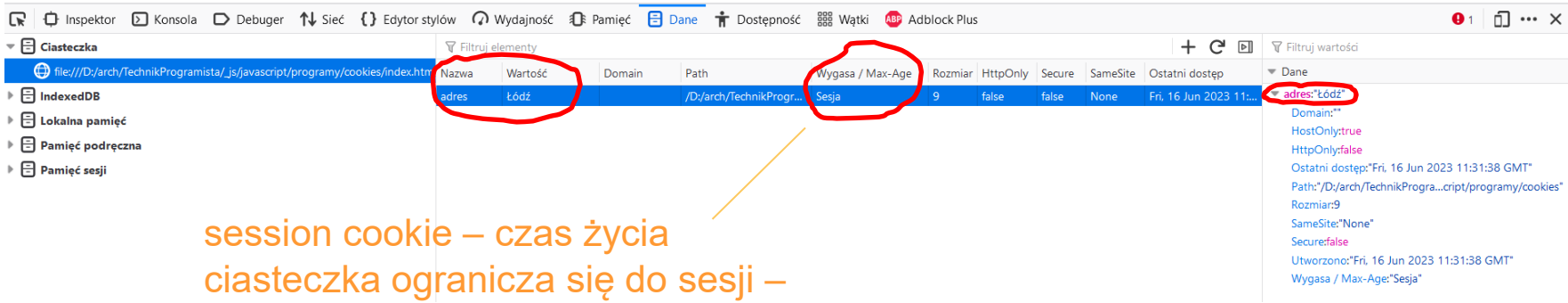
username = Mirek

cookies (ciasteczka)

```
<body>
    <script>
        document.cookie = "adres=Łódź";
    </script>
</body>
```

zapisujemy na komputerze plik cookie z informacją: adres=Łódź

narzędzia developerskie F12 (Firefox)



session cookie – czas życia ciasteczka ogranicza się do sesji – kończy się, gdy opuścimy stronę lub zamkniemy przegladarkę

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

https://developer.mozilla.org/en-US/docs/Web/API/Document/cookie

https://kursjs.pl/kurs/cookies/cookie

cookies (ciasteczka)

```
document.cookie = "adres=Łódź; expires=Fri, 16 Jun 2023 12:14:56 UTC; path=/";
```

cookies (ustawianie - set)

Metoda setTime() ustawia datę i godzinę, dodając (lub odejmując) określoną liczbę milisekund do północy 1 stycznia 1970 r.

czas trwania pliku cookie w ms (30s)

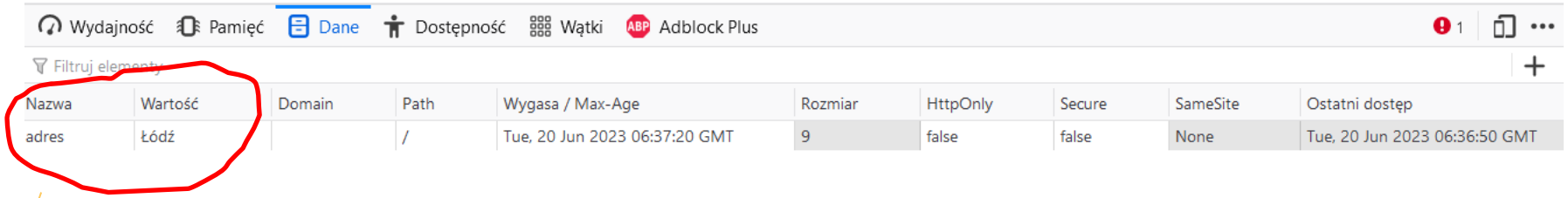
getTime() zwraca liczbę milisekund od 1 stycznia 1970 r. 00:00:00

```
<body>
    <script>
        let cookieTime = 30*1000;

        const date = new Date();

        date.setTime(date.getTime() + cookieTime);

        let expires = "expires="+ date.toUTCString();

        document.cookie = "adres=Łódź; " + expires + "; path=/";
    </script>
</body>
```

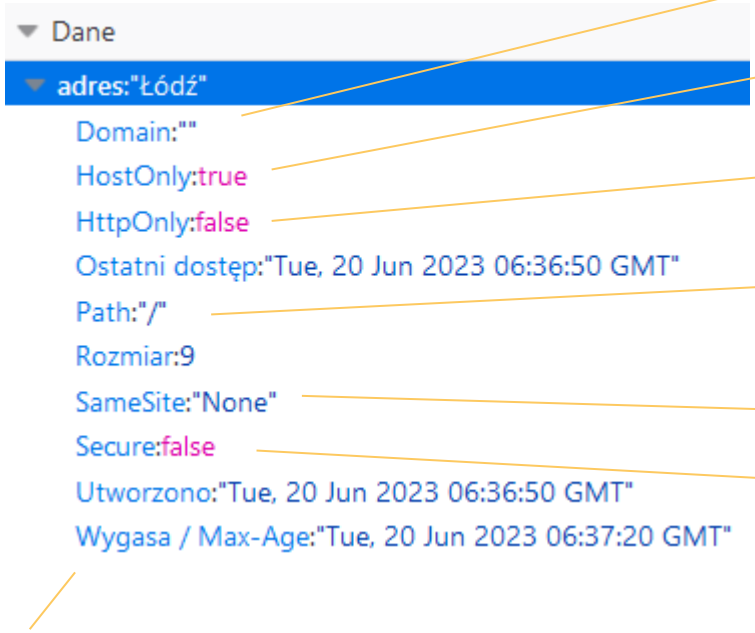
Metoda toUTCString() zwraca obiekt daty jako łańcuch znaków, zgodnie z czasem UTC (Universal Coordinated Time)

ciasteczko adres=Łódź wygaśnie po 30s

cookies (ustawianie - set)



treść zapisanej wiadomości w pliku cookie



atrybut domeny określa, które hosty mogą otrzymać plik cookie

jeśli true: host żądania musi dokładnie odpowiadać domenie pliku cookie

jeśli true: plik cookie jest niedostępny dla skryptów po stronie klienta

atrybut Path wskazuje ścieżkę URL, która musi istnieć w żądanym adresie

dotyczy ochrony przed atakami polegającymi na fałszowaniu żądań między witrynami (cross-site request forgery attacks CSRF)

jeśli true: cookie jest wysyłany do serwera przez protokół HTTPS. Nigdy nie jest wysyłany niezabezpieczonym HTTP (z wyjątkiem lokalnego hosta)

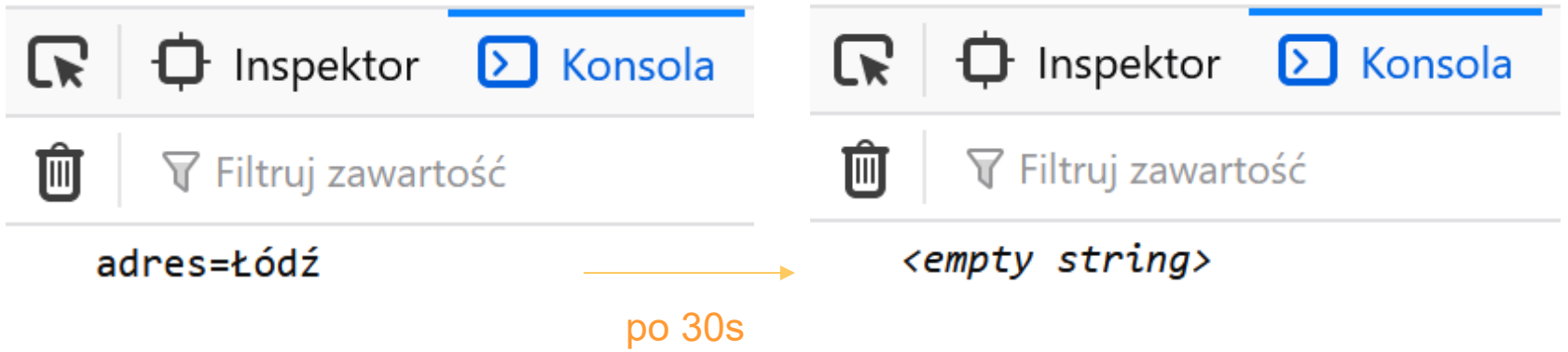
cookie wygasa 30s od utworzenia

https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

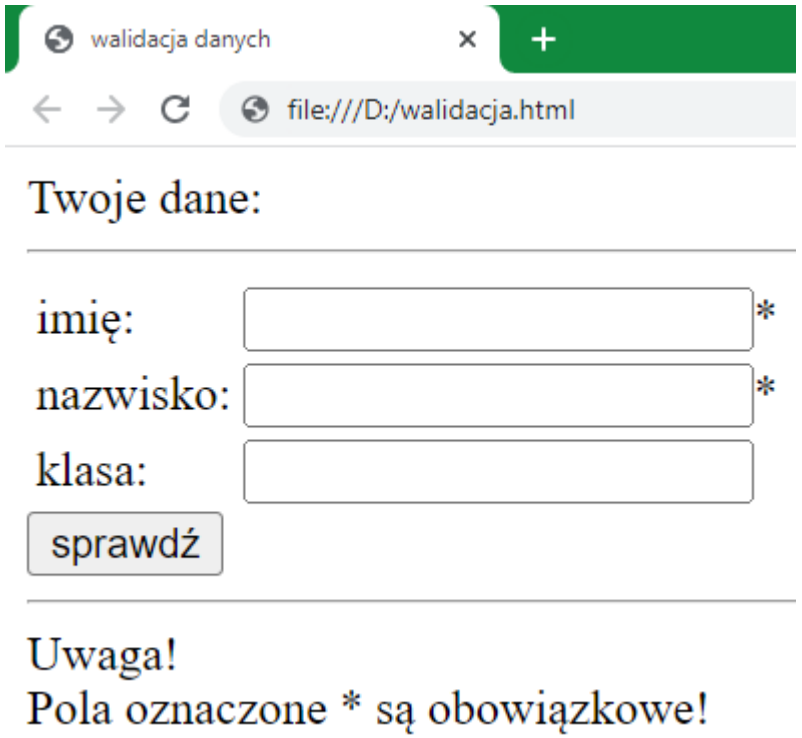
cookies (pobieranie - get)

Funkcja decodeURIComponent() dekoduje pliki cookie

```
<body>
    <script>
        let decodedCookie = decodeURIComponent(document.cookie);
        console.log(decodedCookie);
    </script>
</body>
```



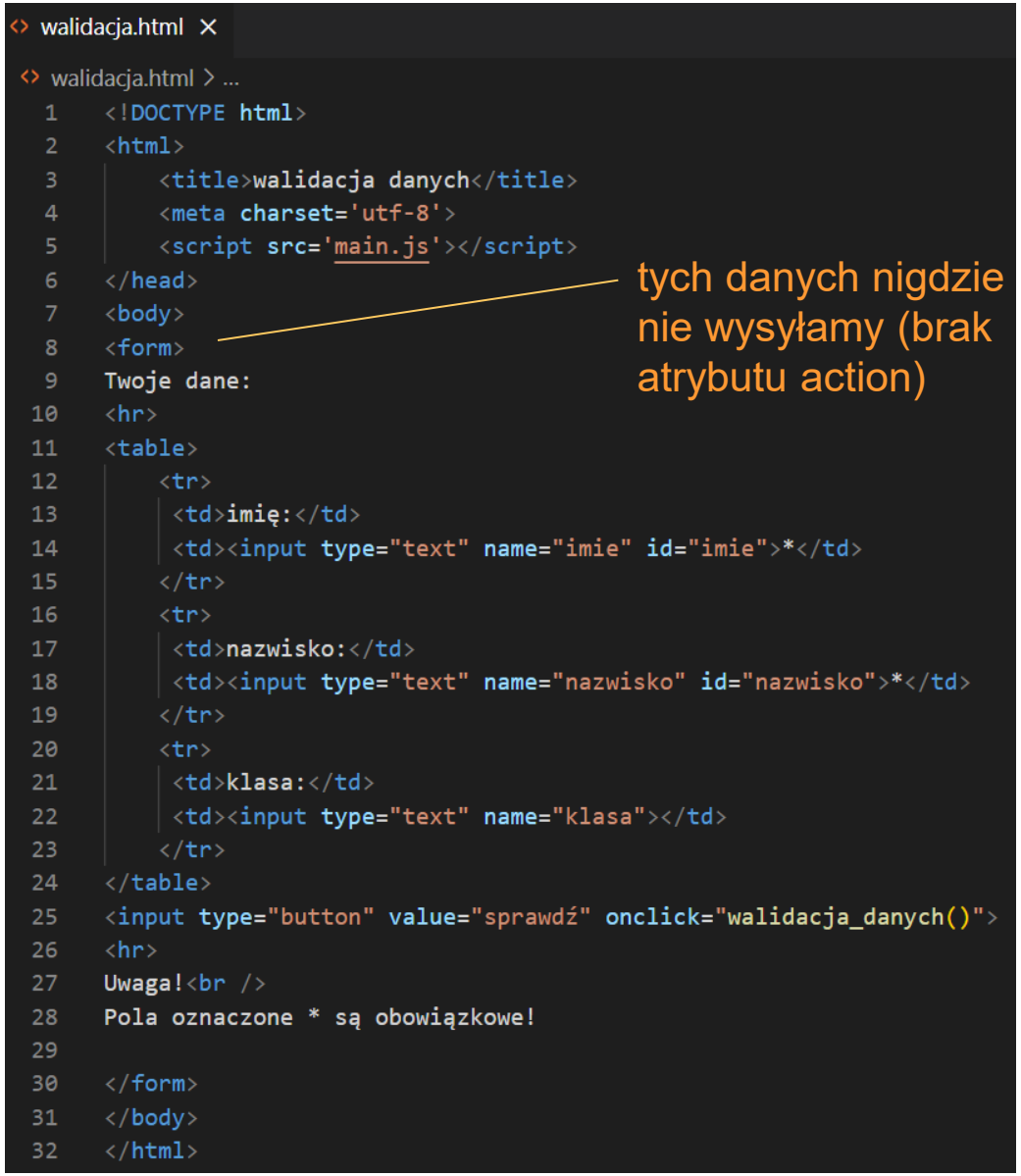
Walidacja danych



sprawdzenie, czy obowiązkowe pola zostały wypełnione

Walidacja danych - formularz

walidacja.html



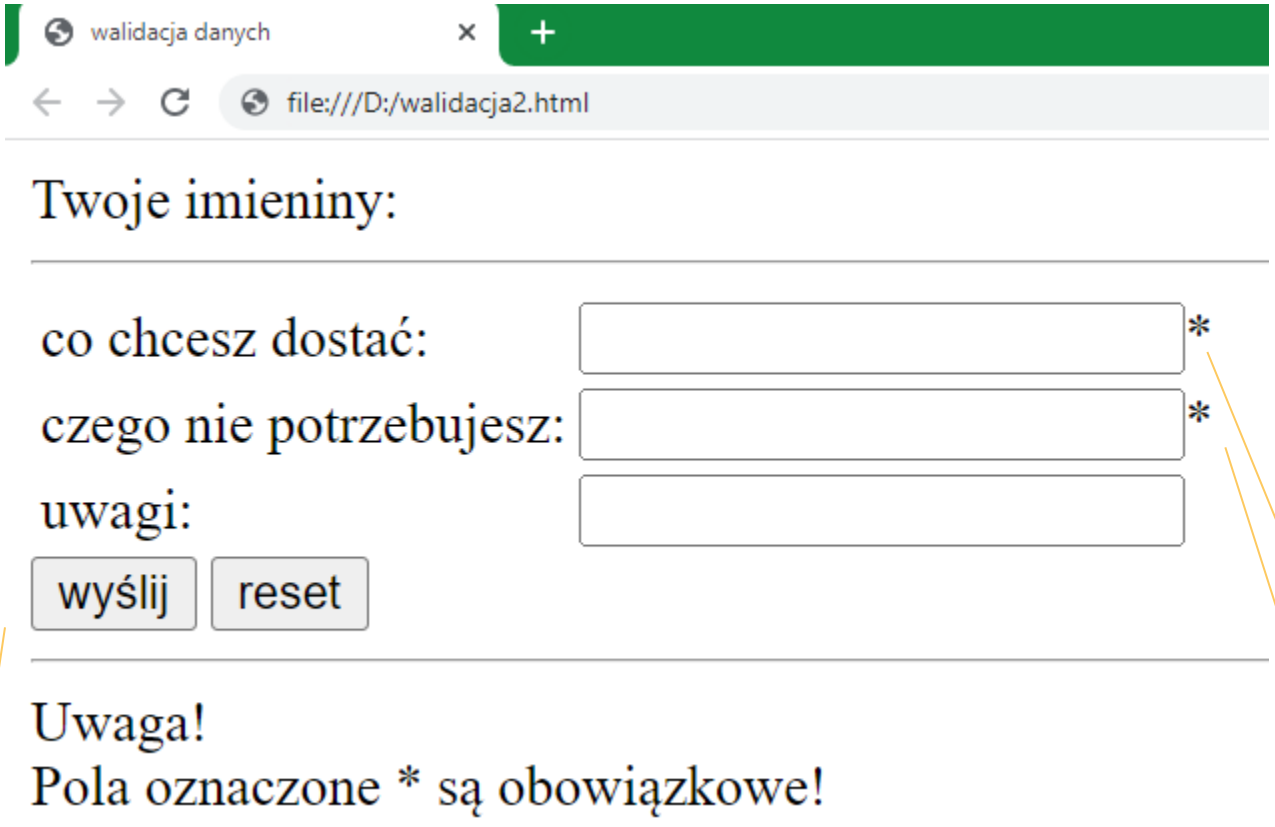
Walidacja danych – skrypt (przykład)

sprawdzenie, czy obowiązkowe pola zostały wypełnione

main.js

```
1  function walidacja_danych (){
2
3      //flaga prawidłowych danych
4      //zakładamy, pola zostały wypełnione prawidłowo
5      var wpis=1;
6
7      //zmienna wypisująca pola, w które nic nie wpisano
8      var braki="";
9
10     //Jeśli nie wpisano nic do pola "imie"
11     if (document.getElementById("imie").value == ""){
12
13         //wpisujemy pole "imie" do braków
14         braki += "imie\n"
15
16         //zerujemy flagę prawidłowych danych
17         wpis=0;
18     }
19     if (document.getElementById("nazwisko").value == ""){
20         braki += "nazwisko\n"
21         wpis=0;
22     }
23
24     //jeśli flaga prawidłowych danych jest prawdą (czyli równa 1)
25     if (wpis)
26       alert("Obowiązkowe pola zostały wypełnione.");
27      //jeżli flaga prawidłowych danych jest fałszem (czyli równa 0)
28      else
29       //wyświetlamy stosowny alert
30       alert ("Nie wypełniono obowiązkowych pól:\n" + braki);
31 }
```

Walidacja danych przed wysłaniem na serwer



te pola nie mogą być puste, długość wpisu musi wynosić co najmniej 3 znaki

po poprawnej walidacji dane są wysłane na serwer

Walidacja danych przed wysłaniem na serwer

walidacja2.html

```
1   <!DOCTYPE html>
2   <html>
3       <title>walidacja danych</title>
4       <meta charset='utf-8'>
5       <script src='main2.js'></script>
6   </head>
7   <body>
8   <form action="walidacja.php" method="post" onsubmit="return  walidacja_danych(this);">
9   Twoje imieniny:
10  <hr>
11  <table>
12      <tr>
13          <td>co chcesz dostać:</td>
14          <td><input type="text" name="prezent" id="prezent">*</td>
15      </tr>
16      <tr>
17          <td>czego nie potrzebujesz:</td>
18          <td><input type="text" name="zlyPrezent" id="zlyPrezent">*</td>
19      </tr>
20      <tr>
21          <td>uwagi:</td>
22          <td><input type="text" name="uwagi" id="uwagi"></td>
23      </tr>
24  </table>
25  <input type="submit" value="wyślij">
26  <input type="reset" value="reset">
27  <hr>
28  Uwaga!<br />
29  Pola oznaczone * są obowiązkowe!
30  </form>
31  </body>
32  </html>
```

po pomyślnej walidacji dane wysyłamy na serwer do pliku walidacja.php

po naciśnięciu przycisku wyślij trigerowane jest zdarzenie onsubmit, które waliduje dane

Walidacja danych przed wysłaniem na serwer

main2.js

```
1   function walidacja_danych (form){
2
3       if (document.getElementById("prezent").value == ""){
4               alert("wpisz jakiś prezent");
5               form.prezent.focus();
6               return false;
7       }else if(document.getElementById("prezent").value.length < 3){
8               alert("nazwa prezentu musi mieć co najmniej 3 znaki");
9               form.prezent.focus();
10              return false;
11      }
12
13      if (document.getElementById("zlyPrezent").value == ""){
14              alert("wpisz czego nie porzebujesz");
15              form.zlyPrezent.focus();
16              return false;
17      }else if(document.getElementById("zlyPrezent").value.length < 3){
18              alert("to, czego nie chcesz musi mieć co najmniej 3 znaki");
19              form.zlyPrezent.focus();
20              return false;
21      }
22
23      return true;
24  }
```

walidacja obowiązkowych pól

przeniesienie fokusu na pole id=zlyPrezent

Input Attributes

Input Restrictions

Here is a list of some common input restrictions:

Attribute	Description
checked	Specifies that an input field should be pre-selected when the page loads (for type="checkbox" or type="radio")
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

HTML Input Types

Here are the different input types you can use in HTML:

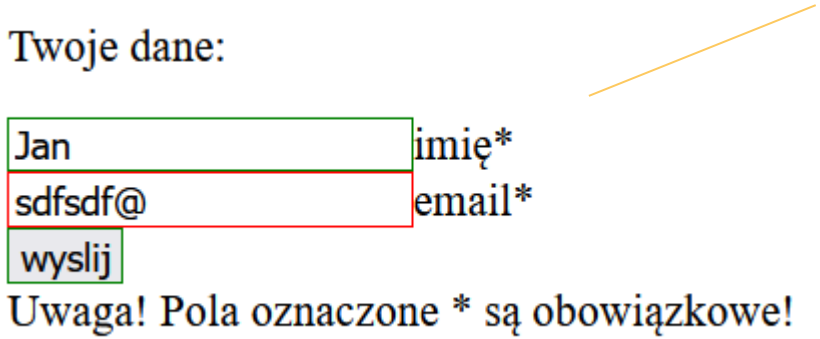
- `<input type="button">`
- `<input type="checkbox">`
- `<input type="color">`
- `<input type="date">`
- `<input type="datetime-local">`
- `<input type="email">`
- `<input type="file">`
- `<input type="hidden">`
- `<input type="image">`
- `<input type="month">`
- `<input type="number">`
- `<input type="password">`
- `<input type="radio">`
- `<input type="range">`
- `<input type="reset">`
- `<input type="search">`
- `<input type="submit">`
- `<input type="tel">`
- `<input type="text">`
- `<input type="time">`
- `<input type="url">`
- `<input type="week">`

niektóre atrybuty formularza mogą być przydatne przy walidacji po stronie przeglądarki (required, step, min, max, pattern)

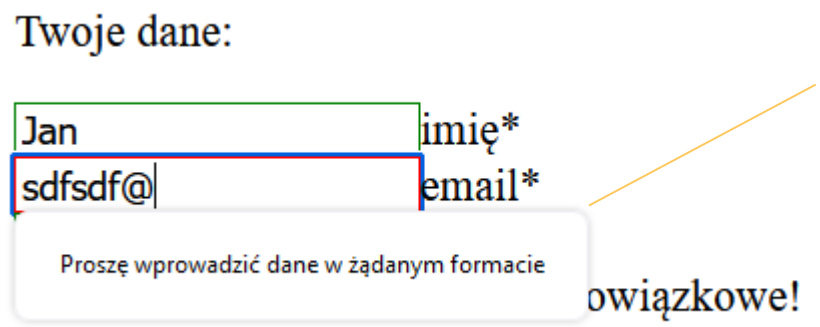
niektóre *Input Types* formularza mogą być przydatne przy walidacji po stronie przeglądarki (numer, email, date, datetime-local)

https://www.w3schools.com/html/html_form_input_types.asp

Walidacja danych z zastosowaniem atrybutów HTML



pola mają ramkę czerwoną, gdy dane zostały wpisane w złym formacie. W innym przypadku mają ramkę zieloną.



gdy dane zostały wpisane w złym formacie ukazuje się stosowany komunikat

Walidacja danych z zastosowaniem atrybutów HTML

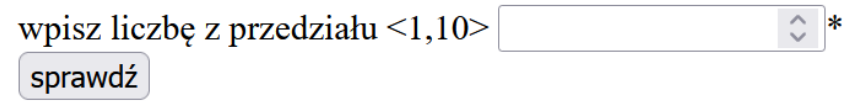
```
<!DOCTYPE html>
<html>
    <title>walidacja danych</title>
    <script>
        function wyslij(){
           document.write("pola wypełnione prawidłowo");
         }
    </script>
    <style>
        input:invalid{
          border: 1px solid red;
        }
        input:valid{
          border: 1px solid green;
        }
    </style>
</head>
<body>
    <form onsubmit="wyslij()">

        <p>Twoje dane:</p>
        <input type="text" pattern="[A-zążęłńóśźżĄĘŁŃÓŚŹŻ]{3,20}" required >imię*<br>
        <input type="text" pattern="[-\w.]+@([A-z0-9][-A-z-9]+\.)+[A-z]{2,4}" required >email*<br>

        <input type="submit" value="wyslij">
        <br>
        Uwaga! Pola oznaczone * są obowiązkowe!

    </form>
</body>
</html>
```

gdy wpisane dane zostaną prawidłowo zwalidowane ukazuje się stosowny komunikat

atrybut *pattern* zawiera wyrażenie regularne, do którego muszą się dostosować wpisywane dane.

atrybut *required* oznacza, że dane muszą być wpisane przed wysłaniem formularza

Walidacja danych z zastosowaniem atrybutów HTML




```html
<!DOCTYPE html>
<html>
    <title>walidacja danych</title>
    <meta charset='utf-8'>
</head>
<body>

wpisz liczbę z przedziału <1,10>
<input type="number" name="liczba" id="liczba" min="1" max="10" required>*
<br>
<button onclick="waliduj()">sprawdź</button>

<br>Pola oznaczone * są obowiązkowe!
<script src='main4.js'></script>
</body>
</html>
```

atrybuty do walidacji liczby

komunikaty walidacji są ukazywane w alercie

```javascript
function waliduj(){

    let liczba = document.getElementById("liczba");

    if (!liczba.checkValidity()) {
        alert(liczba.validationMessage);
    }else{
        alert("pole wypełnione prawidłowo");
    }
}
```

w3schools

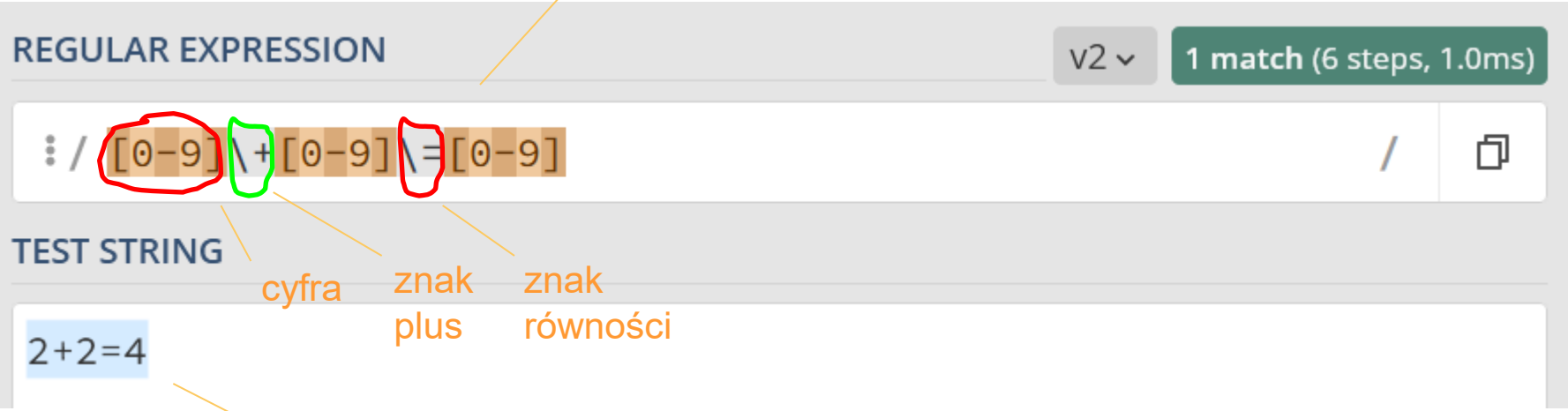
Wyrażenia regularne

Regular Expressions

wzorzec operuje na specjalnych symbolach

wzorzec do którego dopasowujemy ciąg znaków

stwierdzenie dopasowania



cyfra

znak plus

znak równości

ciąg znaków dopasowany do wzorca

piaskownica wyrażeń regularnych

https://www.w3schools.com/jsref/jsref_obj_regexp.asp

https://www.medianauka.pl/wyrazenia-regularne

https://bulldogjob.pl/readme/samouczek-regex-sciagawka-z-przykladami

Wyrażenia regularne

[0-9]\+[0-9]\=[0-9]

wzorzec

2+2=4

3+5=8

0+9=1

ciągi znaków dopasowane do wzorca

~~10+9=0~~

~~1a/b!p~~

ciągi znaków niedopasowane do wzorca

Symbole wzorca

symbol	co oznacza	wzorzec	pasujący do wzorca ciąg znaków
brak	pasuje do każdego ciągu znaków, który ma w sobie słowo **kontra**	kontra	kontrastowy, kontratak, rekontra, kontrabanda
^	początek wzorca	^po	początek, poniedziałek
$	koniec wzorca	ec$	koniec, jeździec
.	pojedynczy znak (dowolny)	m…a	mumia, misia, magia
[…]	Dowolny z podanych w nawiasach znaków [a-z] - wszystkie małe litery. [A-Z] - wszystkie wielkie litery. [A-z] - wszystkie litery. [0-9] - wszystkie cyfry. [A-ząćęłńóśźżĄĘŁŃÓŚŹŻ]– polske znaki	[abc]gat[abc] m[a-z]m[a-z]	agata, dagataewrwre mama, mimi
[^…]	Inny znak niż podany w nawiasach	m[^a-c]m[^a-d] wa[^a]izk[^bcdef]	mimi, memesdfrgf walizka, walizkaaaa

Symbole wzorca

symbol	co oznacza	wzorzec	pasujący do wzorca ciąg znaków
|	lub	mirek | ala (^r|^m)ak	ala, mirek, rak, mak, maks
{2}	dokładnie dwa poprzedzające symbol elementy	a{2} [0-9]{4}	aa 1234, 9879, 2222
{2,}	co najmniej dwa poprzedzające symbol elementy	[a-z]{2,}	as, mirek, waga
{3,4}	od 3 do 4 poprzedzających symbol elementów	[A-Z]{3,4}	ALA, MAMA, TATA
\b	token na początku lub końcu słowa	\bhttps https\b	httpsdsd https xcvhttps (powyżej jeden ciąg znaków) httpsdsd https xcvhttps

Symbole wzorca

symbol	co oznacza	wzorzec	pasujący do wzorca ciąg znaków
\.	Znak specjalny: . * / ? : ^ + \ = | Poprzedzamy ukośnikiem np.: \. oznacza kropkę	*gwiazka [0-9]\+[0-9]\=[0-9]	*gwiazka 2+2=4
(?: …)	Dopasuj wszystko co występuje po dwukropku (non-capturing group – parser nie przypisuje do grupy tego co jest po dwukropku) link	(?:\d{1,3}\.){3} 1-3 cyfry i kropka powtórzone 3 razy	00123.1.12.
(…)	Dopasuj wszystko co występuje w nawiasach (parser przypisze to do oddzielnej grupy)	(ma)	mama ~~smak~~

Symbole flag

flaga	co oznacza
g	global: zwracane są wszystkie pasujące fragmenty w testowanym ciągu znaków
m	multi-line: gdy użyjemy tej opcji ^ i $ będą dopasowywać początek i koniec linii, zamiast odnosić się do całego łańcucha
i	insensitive: nie jest uwzględniana wielkość liter

Klasy znaków

wprowadzone w js

klasa znaku	co oznacza
\d	znak cyfry (to samo co [0-9])
\D	inny znak niż cyfra
\s	znak spacji, tabulacji lub nowego wiersza (biały znak)
\S	znak inny niż spacja, tabulacja lub nowy wiersz (inny niż biały znak)
\w	litera, cyfra lub znak podkreślenia (to samo co [A-Za-z_]
\W	znak, który nie jest literą, cyfrą lub znakiem podkreślenia

przykłady wzorców

wzorzec	co oznacza
\d{2}-\d{3} lub [0-9]{2}-[0-9]{3}	wzorzec kodu pocztowego
[-\w.]+@([A-z0-9][-A-z-9]+\.)+[A-z]{2,4}	wzorzec adresu e-mail
((\bhttps?:\/\/)|(\bwww\.))\S*	wzorzec adresu strony internetowej

sprawdź co znaczą poszczególne symbole

https://regexr.com/

Wyrażenia regularne js

dwa alternatywne sposoby użycia wyrażenia regularnego w js

g, m lub i

let regex = new RegExp(wzorzec, flaga);

let regex = /wzorzec/flaga;

```
RegExp("^[0-9]$");
RegExp("^komputer$");
RegExp("^koMpUteRr$","i");
```

pojedyńcza cyfra

wyraz komputer
pisany małymi literami

wyraz komputer
wielkość liter nie ma
znaczenia

```
/^[0-9]$/
/^komputer$/
/^koMpUteR$/i
```

testowanie wyrazu napis
czy spełnia wymagania
wyrażenia regularnego
(wzorca)

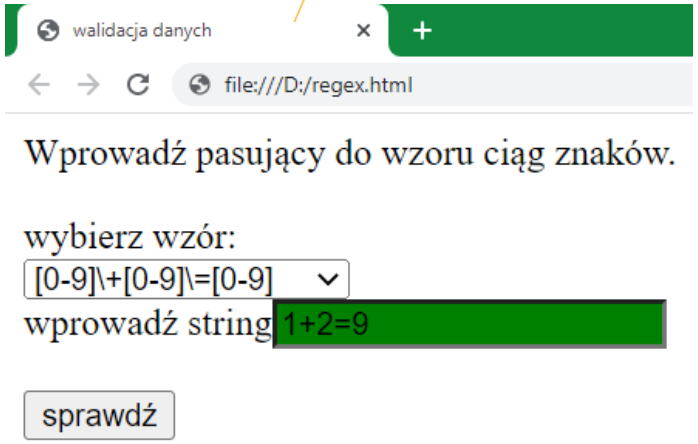
regex.test(”napis”);

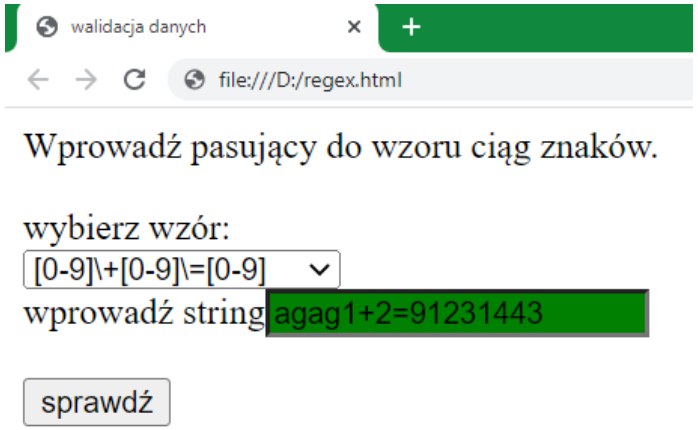
metoda test() zwraca:
true: jeśli wyraz napis pasuje do wzorca
false: w innym przypadku

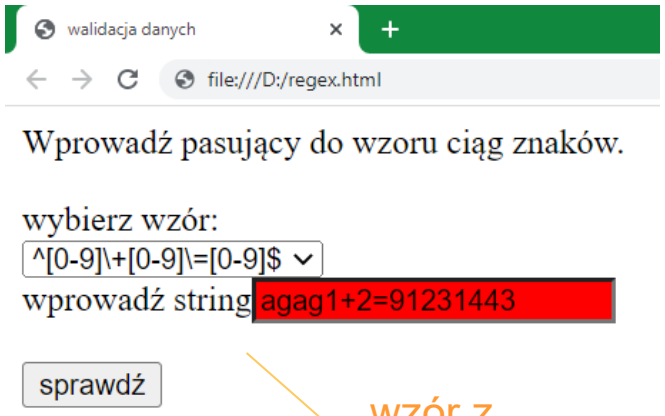
https://javascript.info/regexp-introduction

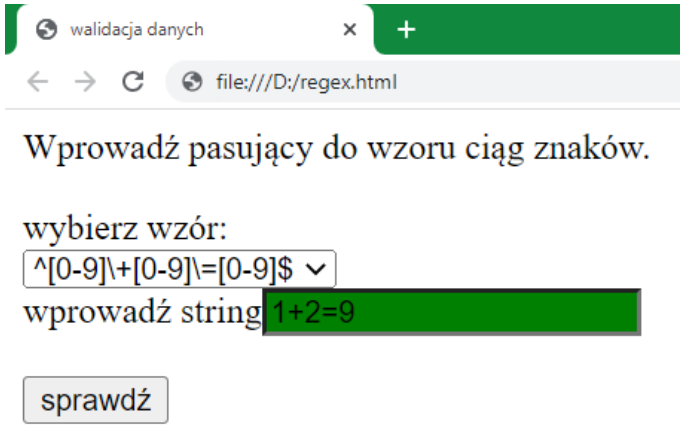
Wyrażenia regularne js

wzór bez ograniczenia początku i końca









wzór z ograniczeniem początku i końca

Wyrażenia regularne js

regex.html

```
<!DOCTYPE html>
<html>
    <title>wyrażenia regularne</title>
    <meta charset='utf-8'>
    <script src='main.js'></script>
</head>
<body>
<form>
    Wprowadź pasujący do wzoru ciąg znaków.<br><br>
    wybierz wzór: <br>
    <select id="opt">
        <option value="reg1">[0-9]\+[0-9]\=[0-9]</option>
        <option value="reg2">^[0-9]\+[0-9]\=[0-9]$</option>
    </select>
    <br>
    wprowadź string<input type="text" id="user-input"><br><br>
    <input type="button" value="sprawdź" onclick="testUserInput()">
</form>
</body>
</html>
```

wybór wzoru

Wyrażenia regularne js

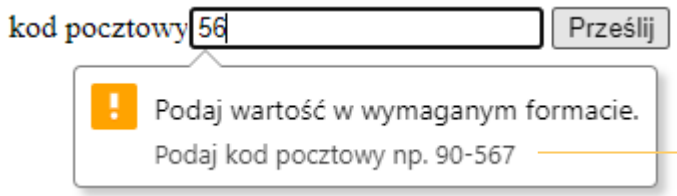
main.js

```
// wzór
let regex;

// test ciągu znaków wprowadzonego przez użytkownika
// czy jest zgodny ze wzorcem (wyrażeniem regularnym)
function testUserInput(){
      let userInput = document.getElementById("user-input");
      let option = document.getElementById("opt");

      // wybór wyrażenia regularnego do testowania
      // w zależności od wyboru użytkownika
      if (option.value == "reg1"){
            regex = /[0-9]\+[0-9]\=[0-9]/;
      }else{
            regex = new RegExp("^[0-9]\\+[0-9]\=[0-9]$");
      }

      // test() zwraca true gdy ciąg pasuje do wzorca
      // w innym przypadku zwraca false
      if(regex.test(userInput.value)){
            userInput.style.background="green";
      }else{
            userInput.style.background="red";
      }
}
```

Wyrażenia regularne HTML

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>Pattern</title>
</head>
<body>
    <form>
        kod pocztowy<input type="text" placeholder="90-567" pattern="\d{2}-\d{3}" title="Podaj kod pocztowy np. 90-567">
        <input type="submit">
    </form>
</body>
</html>
```

wyrażenie regularne można podać w kodzie HTML w formularzu



gdy zostanie wpisana zła wartość pojawi się stosowna informacja

Callbacks

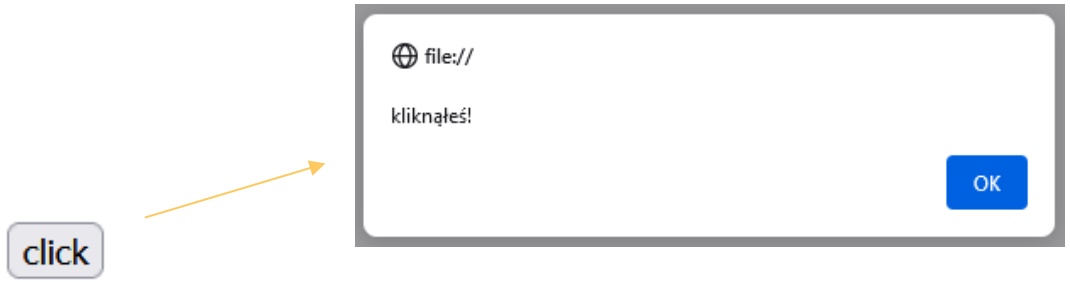
wywołania zwrotne

I will call back later!

```
myBtn = document.getElementById("btn");

// callback
let fun = function(){
    alert("kliknąłeś!");
}

myBtn.addEventListener("click", fun);
```

definicja funkcji zwrotnej

callback to funkcja przekazywana jako argument do innej funkcji, dzięki czemu jedna funkcja może wywołać inną funkcję.



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

https://devmentor.pl/b/callback-functions-w-javascript

Callbacks

```
// callback
let komunikat = function(a){
    myInfo = document.getElementById("info");
    myInfo.innerText = a;
}

let wypisz = function(callbackFun){
    a = "To jest komunikat";
    callbackFun(a)
}

wypisz(komunikat)
```

definicja funkcji zwrotnej

definicja funkcji, której przekazujemy jako argument callback

wywołanie funkcji, której przekazujemy jako argument funkcję ***komunikat*** - callback

To jest komunikat

w wyniku działania funkcji ***wypisz*** na stronie pojawia się komunikat

Callbacks

callbacks są najczęściej używane z funkcjami asynchronicznymi, czyli takimi, które działają równolegle. Dzięki programowaniu asynchronicznemu programy mogą uruchamiać długotrwałe zadania i równolegle wykonywać inne zadania.

setTimeout jest funkcją asynchroniczną

funkcja ***komunikat*** to callback

```
setTimeout(komunikat, 2000);

function komunikat(){
    alert("minęły 2 s");
}
```

definicja funkcji zwrotnej ***komunikat***, która zostanie wywołana po 2s

po 2s pojawia się alert

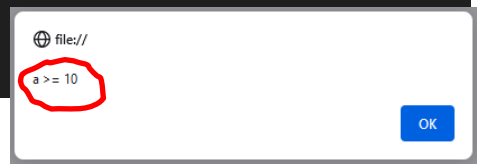
https://www.w3schools.com/js/js_asynchronous.asp

Promises

obietnice

Programy asynchroniczne pisane przy użyciu clallbacks są trudne zarówno do napisania jak i do debugowania. Z tego powodu lepszym rozwiązaniem na tym polu są obietnice (Promises)

```
let myPromise = new Promise(function(myResolve, myReject) {
    // "Producing Code" (May take some time)
    let a = 10;
    if(a >= 10){
        myResolve("a >= 10"); // when successful
    }else{
        myReject("error");  // when error
    }
});

// "Consuming Code" (Must wait for a fulfilled Promise)
myPromise.then(
    function(value) {
        /* code if successful */
        alert(value)
    },
      function(error) {
        /* code if some error */
        alert(error)
    }
);
```



definicja obietnicy przyjmuje jako argument funkcję z dwoma callbackami: myResolve oraz myReject. Po wykonaniu funkcji zostaną one wywołane – pierwsza w przypadku sukcesu lub druga w przypadku błędu. Przekazujemy do nich argumenty, które zostaną przekazane do methody myPromise.then - value, error

metoda then() przyjmuje dwa argumenty w postaci wywołań zwrotnych (callbacks) – jedno w przypadku sukcesu i drugie w przypadku niepowodzenia. Oba argumenty są opcjonalne, więc można dodać wywołanie zwrotne tylko w przypadku powodzenia lub niepowodzenia.

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

Promises

definicja obietnicy

```
const myPromise = new Promise(function(myResolve, myReject) {
    setTimeout(function(){ myResolve("Minęły 3s"); }, 3000);
  });

  myPromise.then(function(value) {
    alert(value)
  });
```

po 3 sekundach od uruchomienia strony zostanie wywołana funkcja anonimowa, która wywoła callback ***myResolve***

Callbacks

```
function getIdea(){
    setTimeout(()=>{
        console.log("1. I'm going to draw a komputer")
    }, 1000);
}
function openPaint(){
    setTimeout(()=>{
        console.log("2. I start Paint")
    }, 500);
}
function draw(){
    setTimeout(()=>{
        console.log("3. I'm drawing a komputer")
    }, 2000);
}
function closePaint(){
    setTimeout(()=>{
        console.log("4. I close Paint")
    }, 200);
}
getIdea();
openPaint();
draw();
closePaint();
```

```
4. I close Paint
2. I start Paint
1. I'm going to draw a komputer
3. I'm drawing a komputer
```

rysowanie komputera

choć funkcje wywoływane są po kolei, ich rezultat działania tego nie odzwierciedla.

w takiej sytuacji z pomocą przychodzą **callbacks**

https://www.youtube.com/watch?v=w7_qAg6EKDE

Callbacks

rysowanie komputera cd.

```
const getIdea = (callback)=>{
    setTimeout(()=>{
        console.log("1. I'm going to draw a komputer");
        callback();
    }, 1000);
}
const openPaint = (callback)=>{
    setTimeout(()=>{
        console.log("2. I start Paint");
        callback();
    }, 500);
}
const draw = (callback)=>{
    setTimeout(()=>{
        console.log("3. I'm drawing a komputer");
        callback();
    }, 2000);
}
const closePaint = (callback)=>{
    setTimeout(()=>{
        console.log("4. I close Paint");
        callback()
    }, 200);
}
getIdea(()=>{
    openPaint(()=>{
        draw(()=>{
            closePaint(()=>{
                console.log("finish!");
            })
        })
    })
})
```

callback hell

po dodaniu callbacs, logi układają się we właściwej kolejności, jednak otrzymaliśmy kod mało czytelny, który trudno debugować (tzw. **callback hell – piekło callbacków**).
Z pomocą przychodzą **Promises (obietnice)**

```
1. I'm going to draw a komputer
2. I start Paint
3. I'm drawing a komputer
4. I close Paint
finish!
```

Promises

```
const getIdea = ()=>{
    return new Promise((resolve, reject)=>{
        setTimeout(()=>{
            console.log("1. I'm going to draw a komputer");
            resolve();
        }, 1000);
    })
}
const openPaint = ()=>{
    return new Promise((resolve, reject)=>{
        setTimeout(()=>{
            console.log("2. I start Paint");
            resolve();
        }, 500);
    })
}
const draw = ()=>{
    return new Promise((resolve, reject)=>{
        setTimeout(()=>{
            console.log("3. I'm drawing a komputer");
            resolve();
        }, 2000);
    })
}
const closePaint = ()=>{
    return new Promise((resolve, reject)=>{
        setTimeout(()=>{
            console.log("4. I close Paint");
            resolve()
        }, 200);
    })
}
```

wywołanie resolve() kończy obietnicę

dzięki Promises otrzymujemy czytelniejszy zapis (łańcuch obietnic)

```
getIdea()
    .then(openPaint)
    .then(draw)
    .then(closePaint)
    .then(()=>{
        console.log("finish!")
    })
```

```
1. I'm going to draw a komputer
2. I start Paint
3. I'm drawing a komputer
4. I close Paint
finish!
```

async function

Słowa kluczowe **async** i **await** umożliwiają pisanie w przejrzystym stylu asynchronicznych, opartych na obietnicach funkcji, unikając konieczności jawnego konfigurowania łańcuchów obietnic.

```
<body>
  <h1></h1>
    <script>
    h = document.getElementsByTagName("h1")[0];
    function marek() {
        return new Promise(resolve => {
          setTimeout(() => {
            resolve('Hallo, tu Marek');
          }, 2000);
        });
      }
      async function asyncCall() {
        h.innerText = 'calling...\n';
        const result = await marek();
        h.innerText += result;
      }
      asyncCall();
    </script>
</body>
```

```
calling...
Hallo, tu Marek
```

funkcja asynchroniczna zwracająca obietnicę

deklaracja funkcji asynchronicznej

słowo **await** powoduje, że funkcje zwracające obietnicę zachowują się tak, jakby były **synchroniczne**, wstrzymując wykonywanie do momentu spełnienia lub odrzucenia zwróconej obietnicy.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function

async function

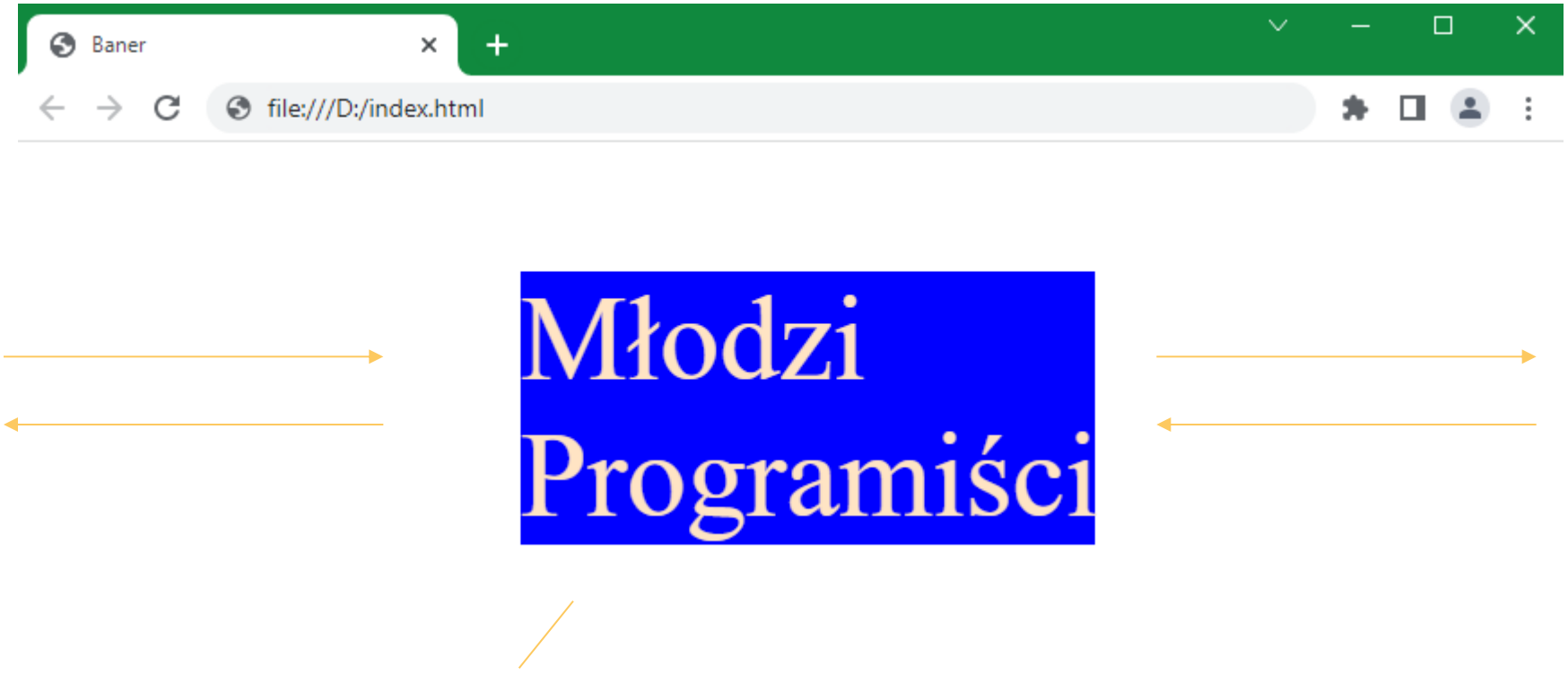
```
<body>
    <h1></h1>
    <script>
        h = document.getElementsByTagName("h1")[0];
        function marek() {
            return new Promise(resolve => {
              setTimeout(() => {
                resolve('Hallo, tu Marek');
              }, 2000);
            });
          }

          function ania() {
            return new Promise(resolve => {
              setTimeout(() => {
                resolve('Hallo, tu Ania');
              }, 2000);
            });
          }

          function janek() {
            return new Promise(resolve => {
              setTimeout(() => {
                resolve('Hallo, tu Janek');
              }, 2000);
            });
          }

          async function asyncCall() {
            h.innerText = 'calling...\n';
            const result1 = await marek();
            h.innerText += result1 + "\n";
            const result2 = await ania();
            h.innerText += result2 + "\n";
            const result3 = await janek();
            h.innerText += result3 + "\n";

          }

          asyncCall();
        </script>
</body>
```

kilka funkcji asynchronicznych zwracających obietnice

funkcje asynchroniczne zachowują się jak synchroniczne – wykonują się jedna po drugiej
(brak łańcucha obietnic)

calling...
Hallo, tu Marek
Hallo, tu Ania
Hallo, tu Janek

animowany baner



animowany baner

index.html

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>Baner</title>
    <link rel="stylesheet" href="main.css">

</head>
<body>
    <p id="p1">Młodzi Programiści</p>
    <script src='main.js'></script>
</body>
</html>
```

animowany napis jest umieszczony w znaczniku <p>

styl animowanego napisu

main.css

```css
#p1{
    position:absolute;
    background-color: blue;
    color: bisque;
    font-size: 100px;
    width: 480px;
}
```

animowany baner

prędkość animowanego napisu

szerokość strony

szerokość napisu

jeśli napis wyjdzie poza obrys strony zmieniamy znak zmiennej speed na przeciwny

metoda setInterval() wywołuje funkcję moveText() co 10 ms (tworzą się klatki animacji)

co 10 ms przesuwamy napis w lewo lub w prawo o wielkość zmiennej speed

przewijany baner

a w Łodzi.Młodzi Programiści to pracownia w Pałacu Młodzieży im. Juliana Tuwim

napis przewija się w lewo

index.html

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>Baner</title>
    <link rel="stylesheet" href="main.css">
</head>
<body>
    <p id="p1"></p>
    <script src='main.js'></script>
</body>
</html>
```

napis jest umieszczony w znaczniku <p>

styl napisu

main.css

```css
#p1{
    background-color: blue;
    color: bisque;
    font-size: 40px;
    width:1360px;
    margin-left: auto;
    margin-right: auto;
}
```

przewijany baner

main.js

```
1   let p1 = document.getElementById("p1");
2
3   let i = 0;
4   let banner = "Młodzi Programiści to pracownia w Pałacu Młodzieży im. Juliana Tuwima w Łodzi.";
5
6
7   function moveText(){
8       p1.innerHTML = banner.substring(i)  +  banner.substring(0,i);
9       i++;
10      if(i==banner.length){
11          i=0;
12      }
13  }
14
15  window.setInterval(moveText,1000);
```

wycina substring od i-tej literki do końca napisu

wycina substring od początku napisu do i-tej literki

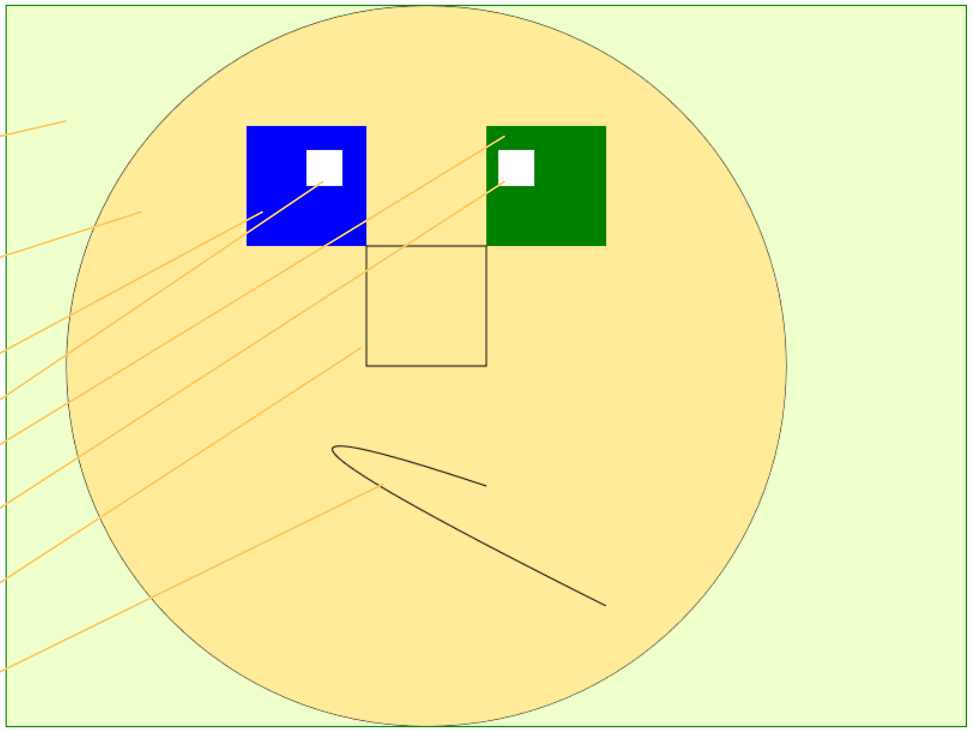
metoda setInterval() wywołuje funkcję moveText() co 1s (tworzą się klatki animacji)

w każdej klatce animacji wyświetlamy stosowny napis korzystając z metody substring() oraz inkrementujemy zmienną i (miejsce podziału napisu)

canvas

definiujemy płótno na rysunek
o wymiarach 800x600

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="800" height="600"
style="border:1px solid green">
Your browser does not support the canvas element.
</canvas>

<script>
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");

ctx.fillStyle = "#eeffcc";
ctx.fillRect(0,0,800,600);


ctx.fillStyle = "#ffeb99";
ctx.beginPath();
ctx.arc(350, 300, 300, 0, 2 * Math.PI);
ctx.stroke();
ctx.fill();

ctx.fillStyle = "#0000ff";
ctx.fillRect(200,100,100,100);
ctx.clearRect(250, 120, 30, 30);
ctx.fillStyle = "green";
ctx.fillRect(400,100,100,100);
ctx.clearRect(410, 120, 30, 30);

ctx.strokeRect(300,200,100,100);

ctx.beginPath();
ctx.moveTo(400, 400);
ctx.quadraticCurveTo(100,300, 500, 500);
ctx.stroke();

</script>
</body>
</html>
```



canvas tutorial

canvas reference

canvas reference

canvas animacja

```html
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="800" height="600"
style="border:1px solid green">
Your browser does not support the canvas element.
</canvas>

<script>
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
var x = 0;
const auto = new Image();
auto.src="./auto.png";

function face(){
    ctx.fillStyle = "#eeffcc";
    ctx.fillRect(0,0,800,600);


    ctx.fillStyle = "#ffeb99";
    ctx.beginPath();
    ctx.arc(350, 300, 300, 0, 2 * Math.PI);
    ctx.stroke();
    ctx.fill();

    ctx.fillStyle = "#0000ff";
    ctx.fillRect(200,100,100,100);
    ctx.clearRect(250 - x, 170, 30, 30);
    ctx.fillStyle = "green";
    ctx.fillRect(400,100,100,100);
    ctx.clearRect(410 - x, 170, 30, 30);

    ctx.strokeRect(300,200,100,100);

    ctx.beginPath();
    ctx.moveTo(400, 400);
    ctx.quadraticCurveTo(100,300+2*x, 500, 500);
    ctx.stroke();

    ctx.drawImage(auto, 300-10*x,500);

    x++;
    if(x > 50){
        x = -x;
    }
}
window.setInterval(face, 10);
</script>
</body>
</html>
```

zmienna x do zmiany położenia oczu, auta i ust

konfiguracja obrazka auta

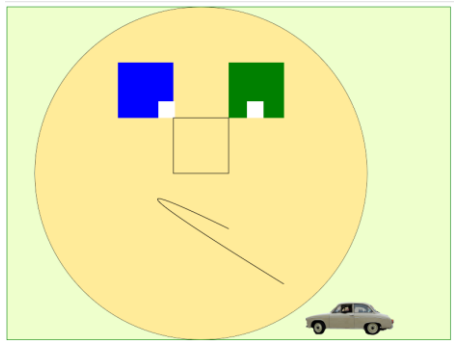
zmiana położenia oczu

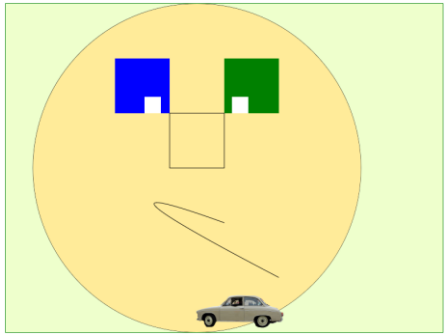
zmiana położenia ust

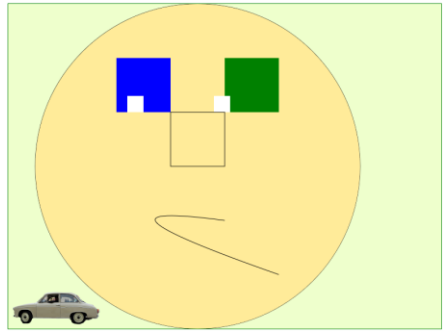
wstawienie obrazka auta

inkrementacja i konfiguracja x

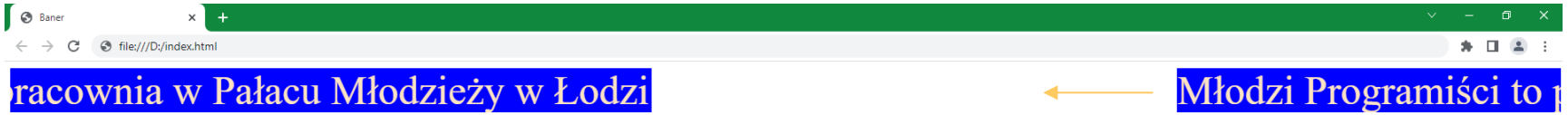
konfiguracja pętli klatek animacji (klatka co 10ms)







animowany baner 2



obrazek napis.png
przesuwa się w lewo

animowany baner 2

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>Baner</title>
</head>
<body>
    <script>
        document.write(
            "<canvas id='myCanvas' width=" + (window.innerWidth - 10) + "height='60px'> \
            Your browser does not support the HTML5 canvas tag.\
            </canvas>"
        )
    </script>
    <script src='main.js'></script>
</body>
</html>
```

używamy js aby ustawić szerokość canvas na trochę mniejszą niż szerokość strony

definiujemy canvas

animowany baner 2

JS main.js

```
1   var canvas = document.getElementById("myCanvas");
2   var ctx = canvas.getContext("2d");
3   const img = new Image();
4   img.src = './napis.png';
5
6   let i = 0;
7
8   function moveText() {
9       ctx.clearRect(0, 0, window.innerWidth, 60);
10      ctx.drawImage(img, i, 0);
11      ctx.drawImage(img, i + window.innerWidth, 0);
12      i -= 1;
13
14      if (Math.abs(i) > window.innerWidth){
15          i = 0;
16      }
17  };
18
19
20  window.setInterval(moveText,10);
```

definiujemy zmienną img z obrazkiem napis.png

współrzędna x lewego boku obrazka

czyścimy canvas

obrazek z napisem wstawiamy równocześnie na początku i na końcu canvas

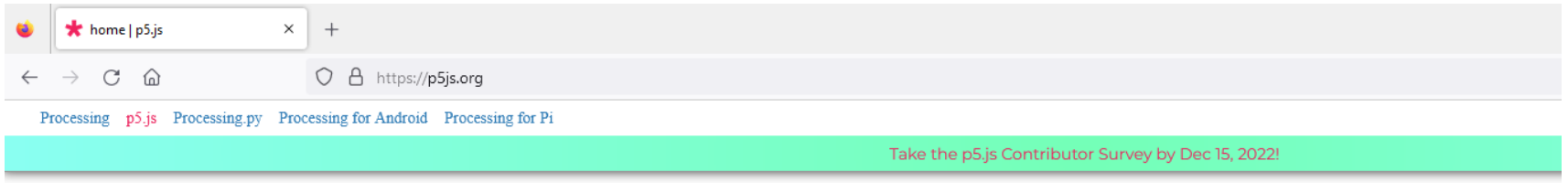
dekrementacja x

gdy współrzędna x osiągnie wielkość równą szerokości ekranu resetujemy ją i cykl rysowania zaczyna się od początku

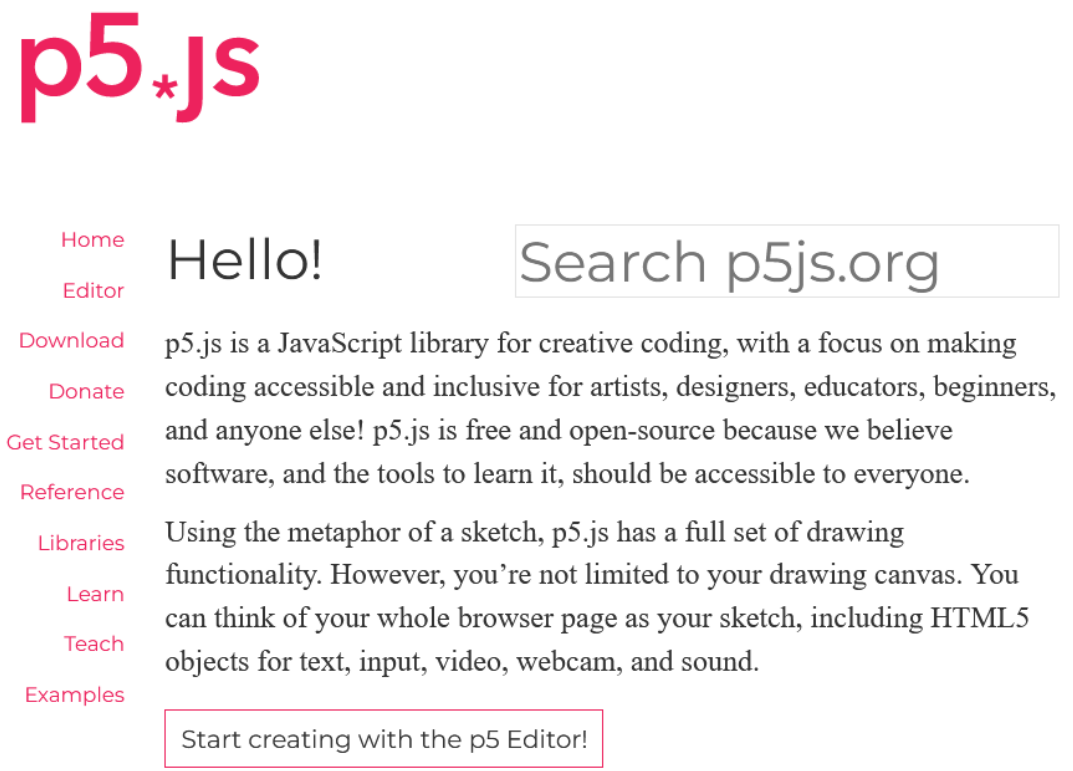
metoda setInterval() wywołuje funkcję moveText() co 10 ms (tworzą się klatki animacji)

p5.js

https://p5js.org/



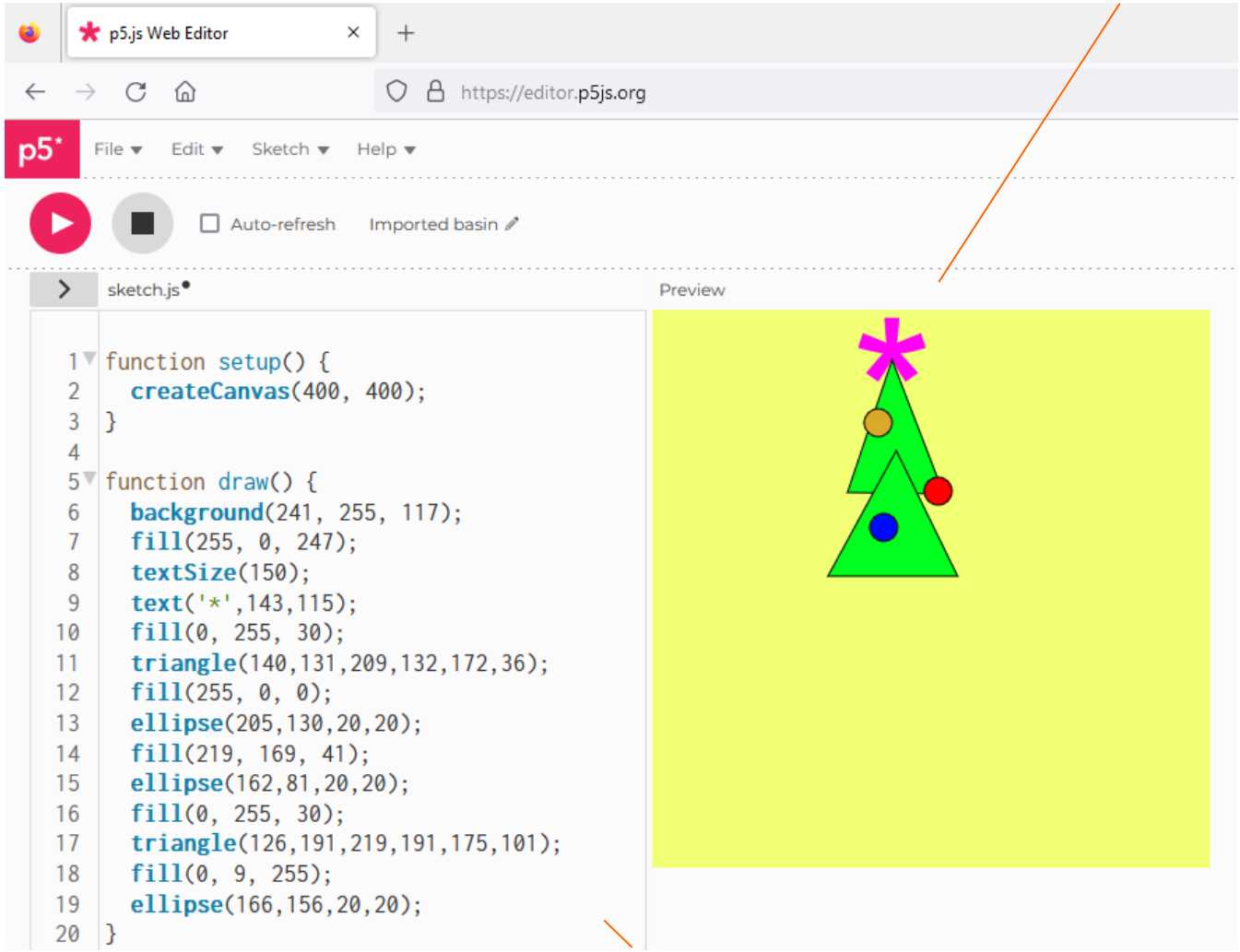
p5.js to darmowa biblioteka open-source napisana w języku JavaScript służąca do kreatywnego kodowania.



p5.js

https://editor.p5js.org/

dorysuj brakującą część choinki oraz prezenty (może być też Mikołaj)



plik sketch.js

p5.js

funkcja setup() wykonuje się jednokrotnie po załadowaniu strony

płótno, na którym rysujemy

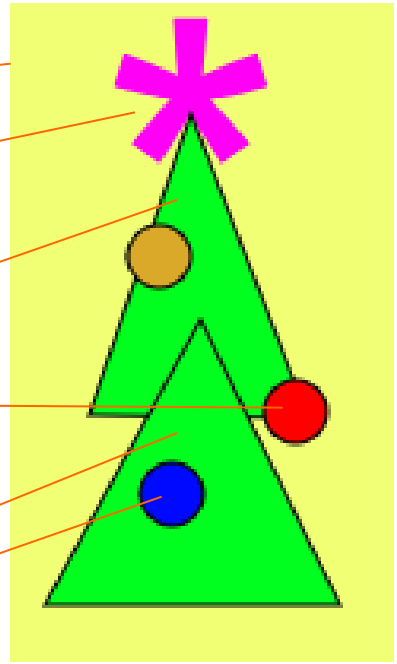
kolor tła

kubełek wypełniający kolorem wszystko co jest poniżej

trójkąt

bombki

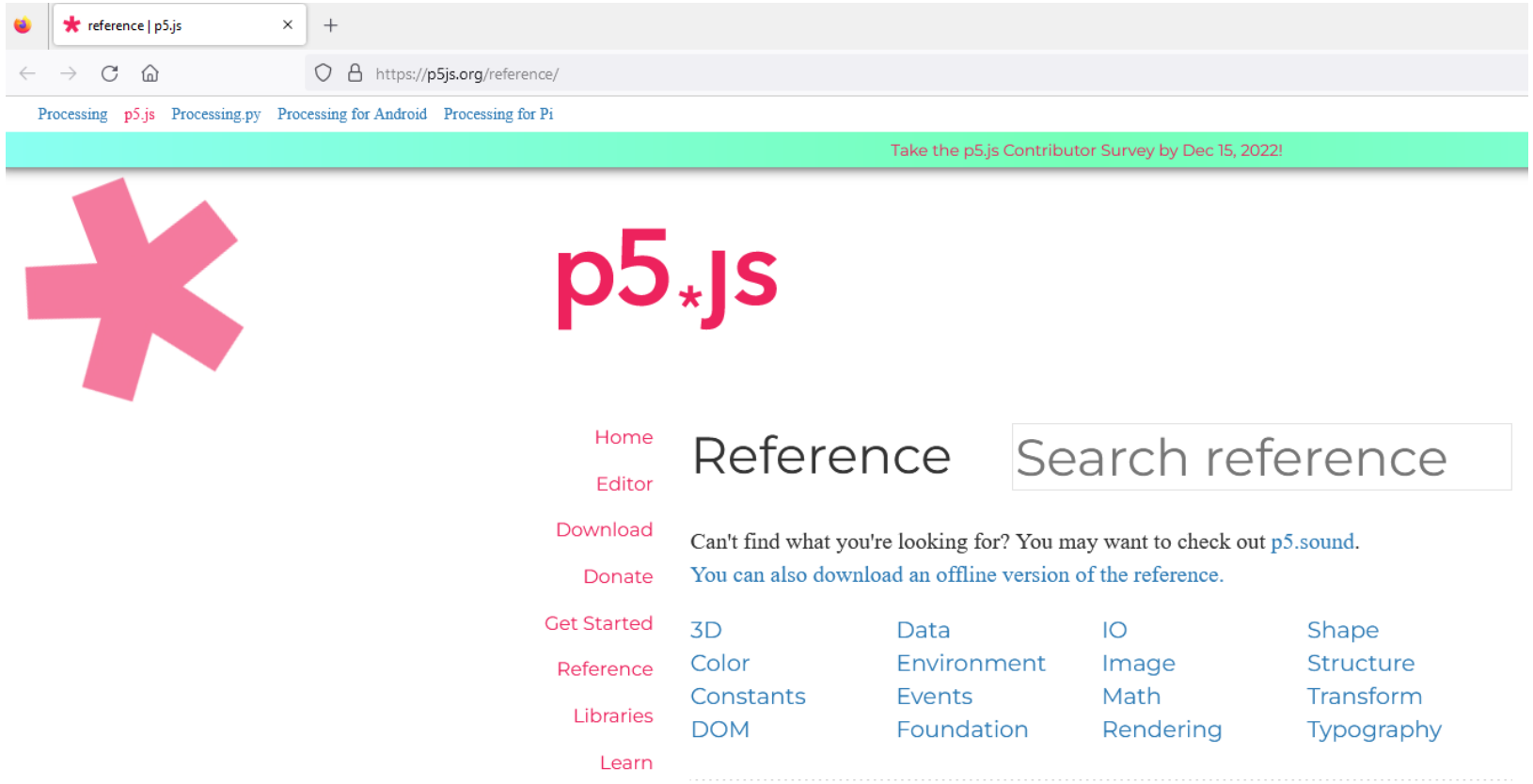
```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(241, 255, 117);
  fill(255, 0, 247);
  textSize(150);
  text('*',143,115);
  fill(0, 255, 30);
  triangle(140,131,209,132,172,36);
  fill(255, 0, 0);
  ellipse(205,130,20,20);
  fill(219, 169, 41);
  ellipse(162,81,20,20);
  fill(0, 255, 30);
  triangle(126,191,219,191,175,101);
  fill(0, 9, 255);
  ellipse(166,156,20,20);
}
```



funkcja draw() wykonuje się w nieskończonej pętli

p5.js Reference

https://p5js.org/reference/

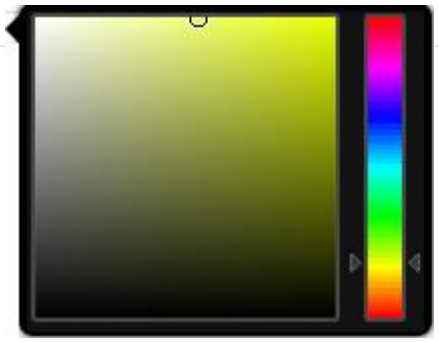


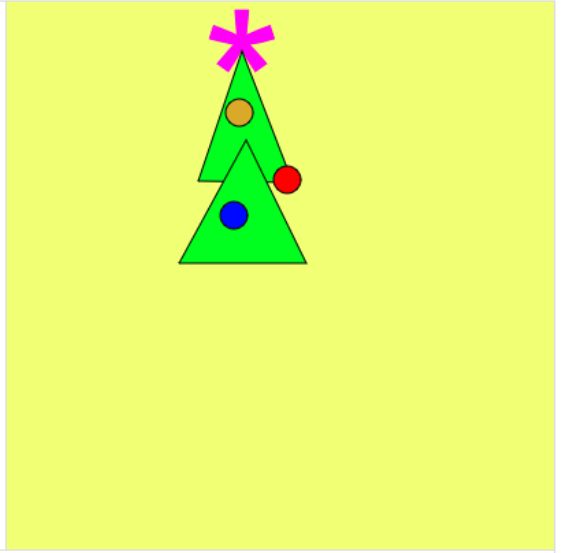
p5.js
Khan Academy

Khan Academy

możliwość wyboru koloru z palety

```
1  background(241, 255, 117);
2  fill(255, 0, 247);
3  textSize(150);
4  text('*',143,115);
5  fill(0, 255, 30);
6  triangle(140,131,209,132,172,36);
7  fill(255, 0, 0);
8  ellipse(205,130,20,20);
9  fill(219, 169, 41);
10 ellipse(170,81,20,20);
11 fill(0, 255, 30);
12 triangle(126,191,219,191,175,101);
13 fill(0, 9, 255);
14 ellipse(166,156,20,20);
15
```

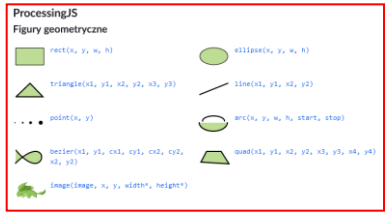




możliwość łatwego przesunięcia punktu

0:00/6:08 Obejrzyj na YouTube Spin-off

O nas Dokumentacja Spin-offy Transkrypcja



edytor p5.js z Khan Academy ma kilka zalet

p5.js umieszczenie rysunku na stronie

EXPLORER

CHOINKA0

choinka.html

sketch.js

choinka.html

```
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/p5@1.5.0/lib/p5.js"></script>
    <script src="sketch.js"></script>
  </head>
  <body>
    <h1>choinka</h1>
  </body>
</html>
```

link do biblioteki p5.js

EXPLORER

CHOINK...

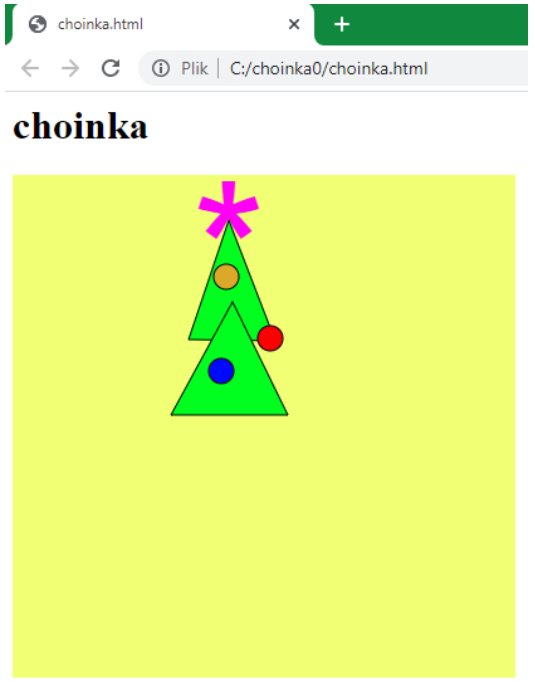
choinka.html

sketch.js

sketch.js

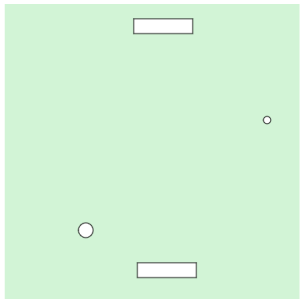
```
function setup() {
  createCanvas(400, 400);
}

function draw() {
    background(241, 255, 117);
    fill(255, 0, 247);
    textSize(150);
    text('*',143,115);
    fill(0, 255, 30);
    triangle(140,131,209,132,172,36);
    fill(255, 0, 0);
    ellipse(205,130,20,20);
    fill(219, 169, 41);
    ellipse(170,81,20,20);
    fill(0, 255, 30);
    triangle(126,191,219,191,175,101);
    fill(0, 9, 255);
    ellipse(166,156,20,20);
}
```

kod choinki

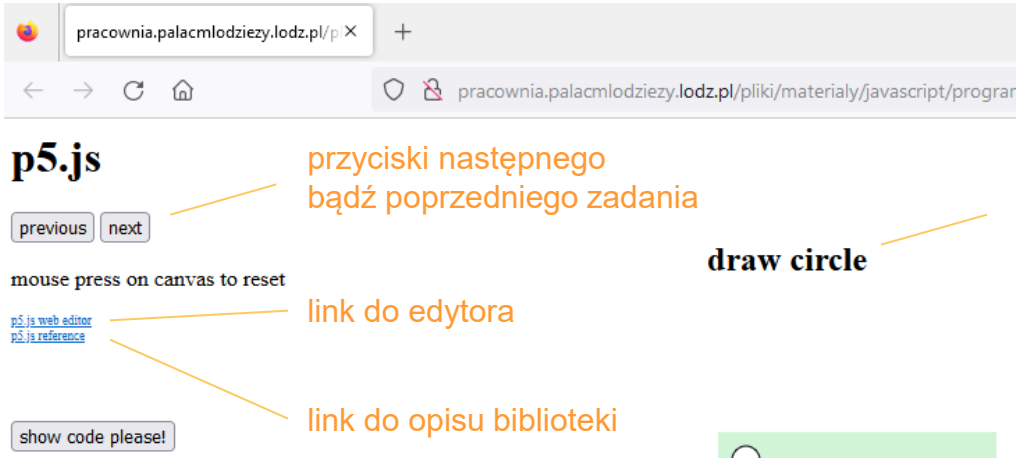


https://p5js.org/get-started/

p5.js
Ping Pong tutorial



efekt końcowy po wykonaniu zadań z tutoriala – gra Ping Pong na dwie osoby z latającą przeszkodą



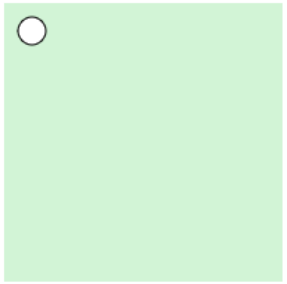
przyciski następnego bądź poprzedniego zadania

link do edytora

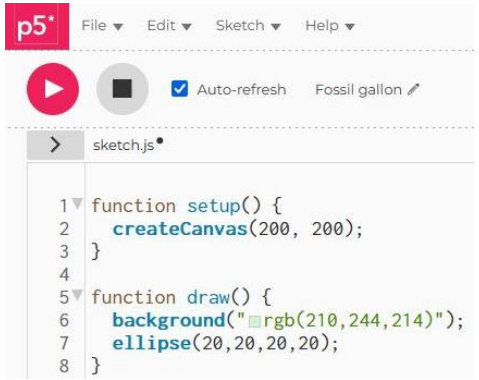
link do opisu biblioteki

treść zadania, które wykonujemy w edytorze pisząc odpowiedni kod

draw circle



końcowy wygląd zadania



jeśli nie uda się napisać kodu można zajrzeć w odpowiedź

Ping Pong tutorial

Typowe błędy

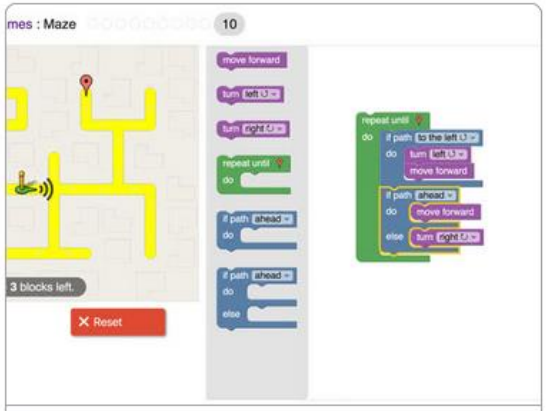
```
<html>
<head>
<script type="tekst/javascript">
function
funcion mnozenie(a,b) ;
{
return a*b;
}
</script>
</head>

<body>
<script type="text/javascript">
document.write("12*5=" + mnożenie(12,5));
</script>
</body>
</html>
```

text

z

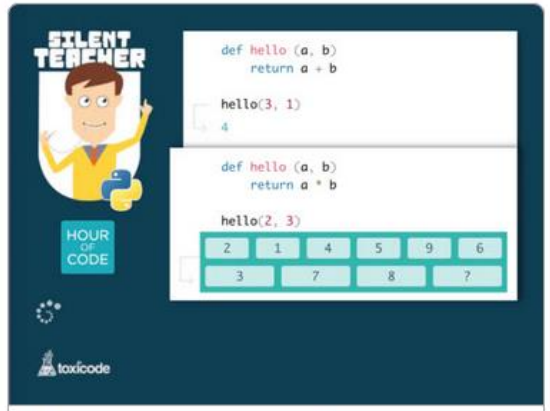
Zjadając ważne znaki
generujesz sobie braki!

Godzina Kodowania

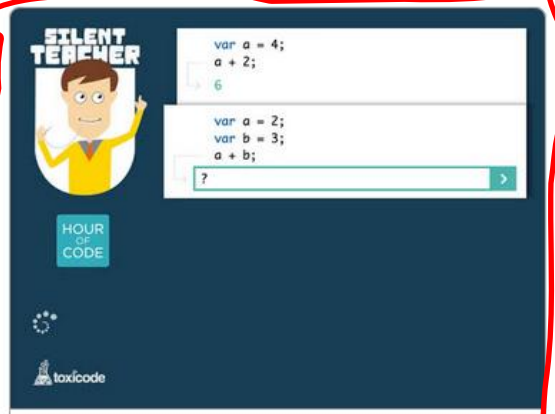


Gry w Blockly
Klasy 6+ (od 11 lat) | Bloki

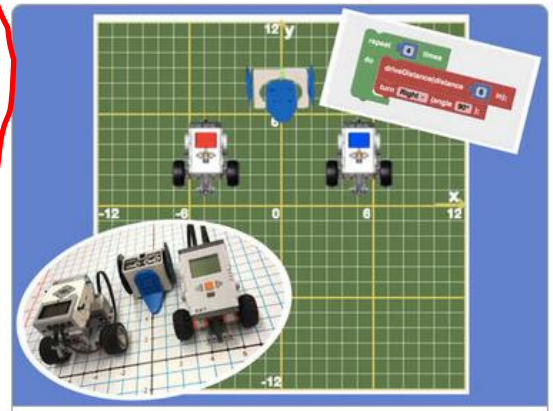
Galaxy Game Jam
Klasy 6+ (od 11 lat) | Bloki | Android



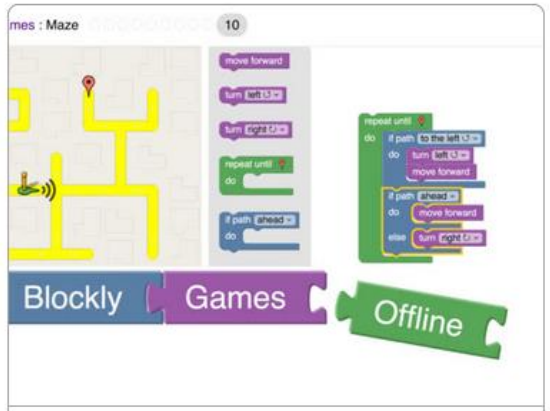
Odkryj Pythona z Silent Teacher
Klasy 6+ (od 11 lat) | Python



Milczący nauczyciel
Klasy 6+ (od 11 lat) | JavaScript



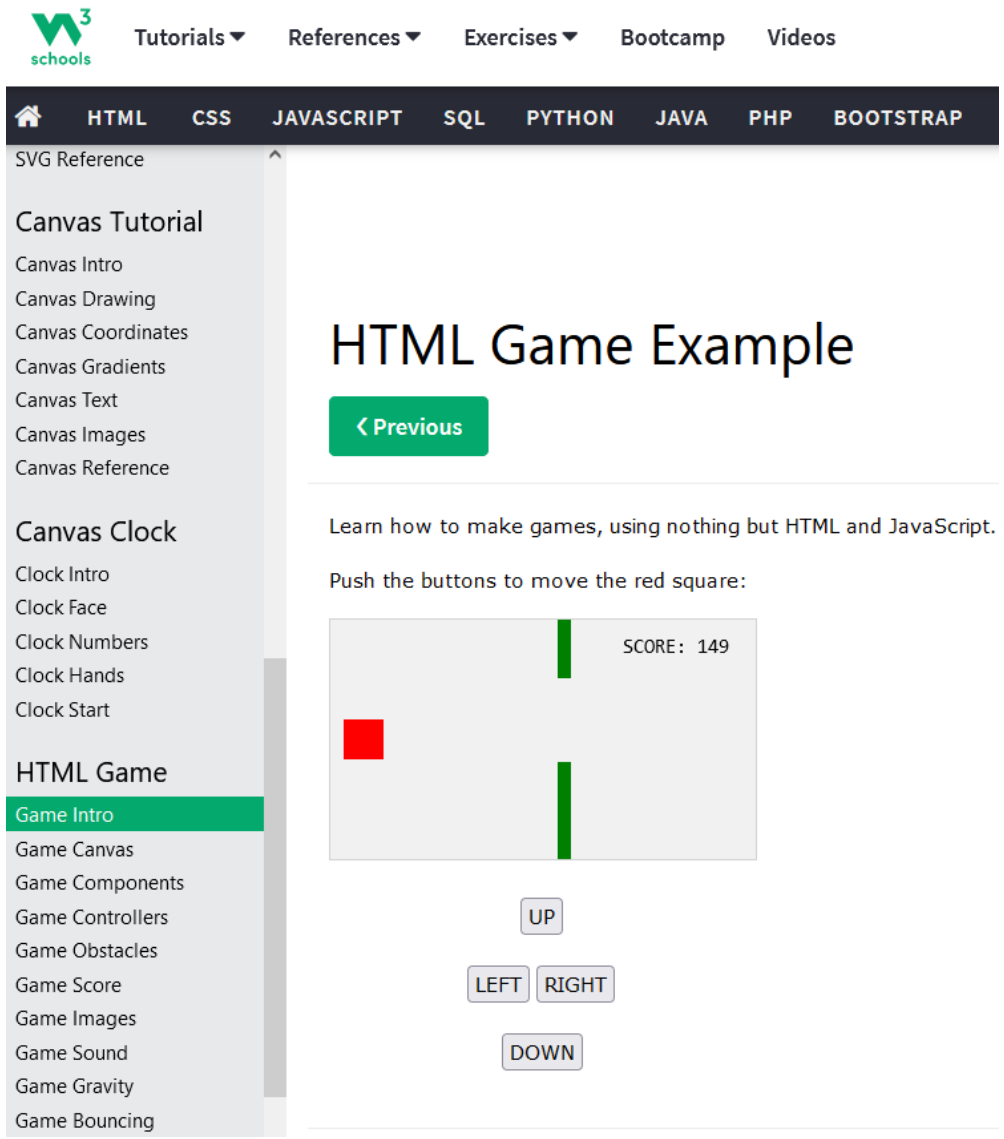
Poznaj algebrę z RoboBlockly
Klasy 6+ (od 11 lat) | Bloki, Ch/C/C++



Blockly
Klasy 6+ (od 11 lat) | Bloki

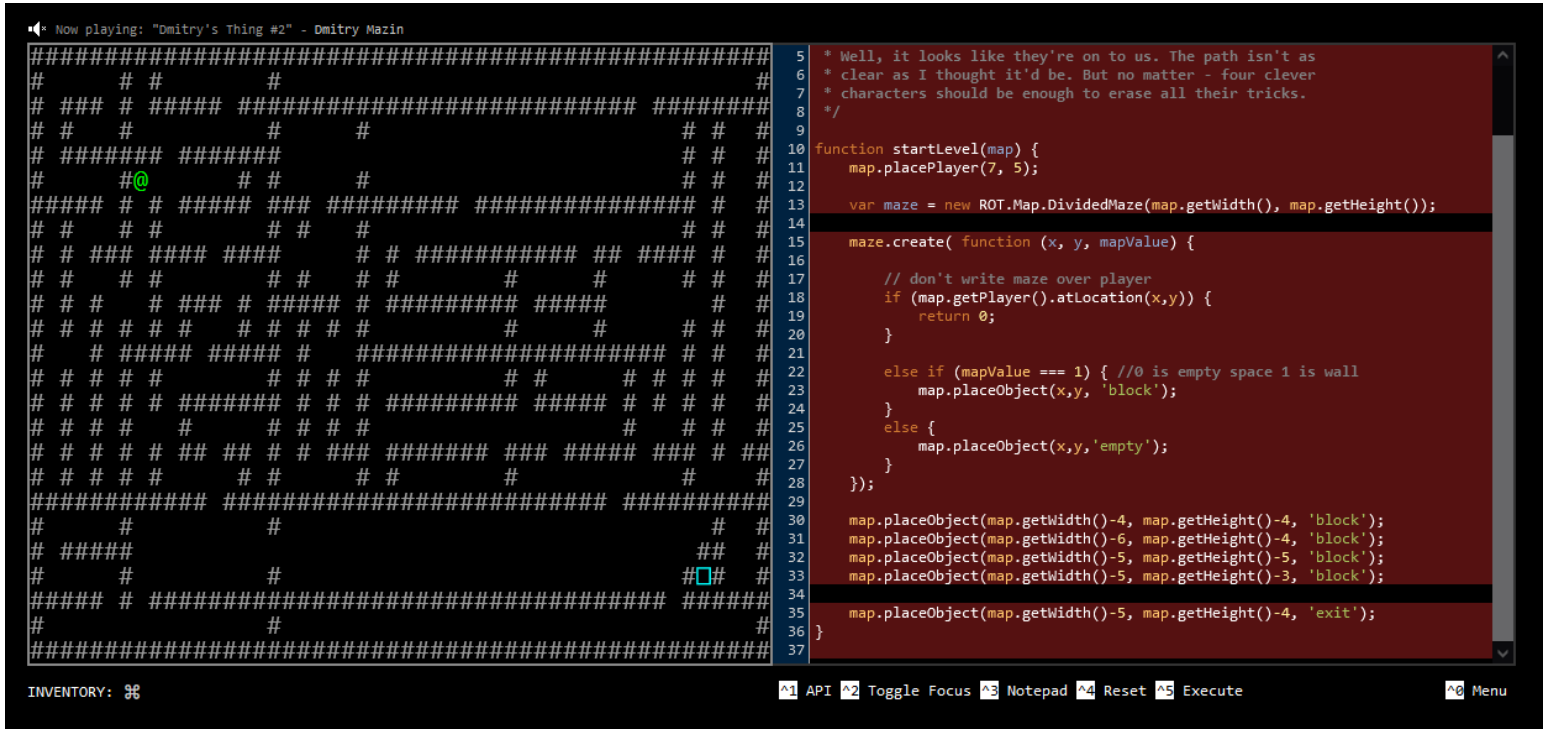
https://silentteacher.toxicode.fr/hourofcode

HTML JS Game Tutorial



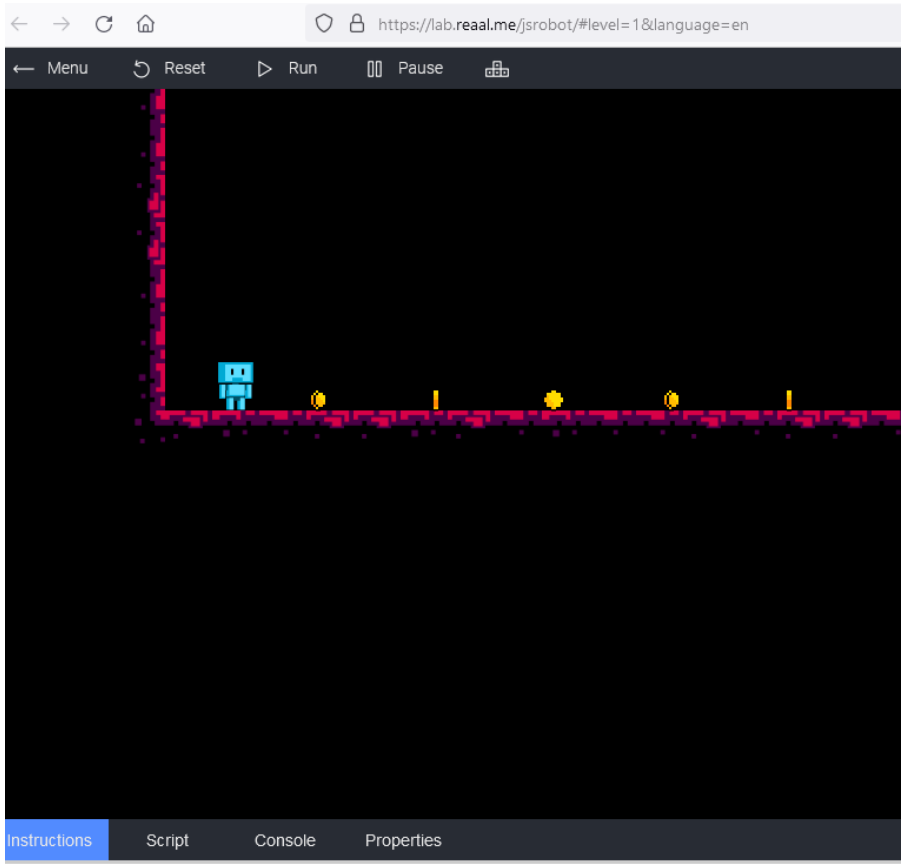
https://www.w3schools.com/graphics/game_intro.asp

Game



https://alexnisnevich.github.io/untrusted/

Game



• Controlling the Robot

The objective of the game is to reach the flag at the end of each level. The robot can take damage an

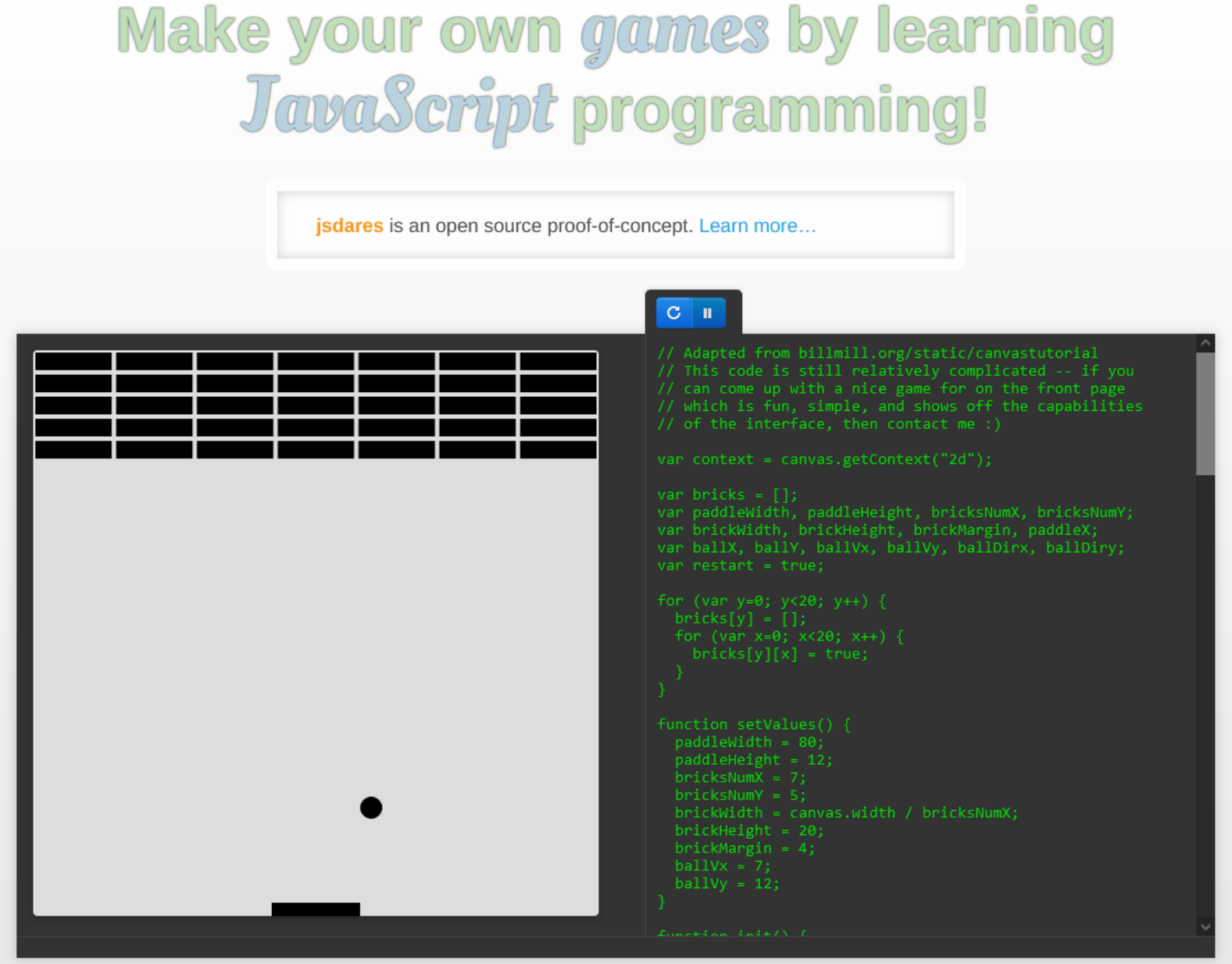
The robot has multiple **actions**: `move`, `jump`, `shoot`, `turn` and `wait`.

The `move` action takes an **amount** between **-40** and **40**. A positive amount moves the robot to the rig
The `jump` action also an **amount**, between **-10** and **10**, and as before a positive amount tells the robo

https://lab.reaal.me/jsrobot/#level=1&language=en

Game

https://jsdares.com/?



Game

Wow, you did everything! Congratulations, nice work! A lot of these are really hard. I'm impressed you finished! I hope you enjoyed it!

```
var pane = document.getElementById('pane');
var s = 3;

pane.onmousemove = function(evt) {
  c.fillStyle = randomRGBA();
  var x = evt.clientX;
  var y = evt.clientY;
  c.fillRect(x - s / 2, y - s / 2, s, s);};

function randomRGBA() {
  var r = randInt(255);
  var g = randInt(255);
  var b = randInt(255);
  var a = Math.random();
  var rgba = [r,g,b,a].join(",");
  return "rgba(" + rgba + ")";
}
function randInt(limit) {
  var x = Math.random() * limit;
  return Math.floor(x);
}
```

RESET

The End

BACK

https://www.crunchzilla.com/code-maven

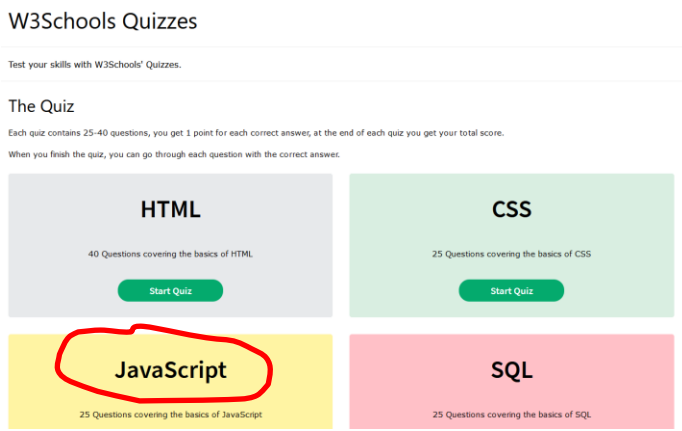
Game

https://play.elevatorsaga.com/

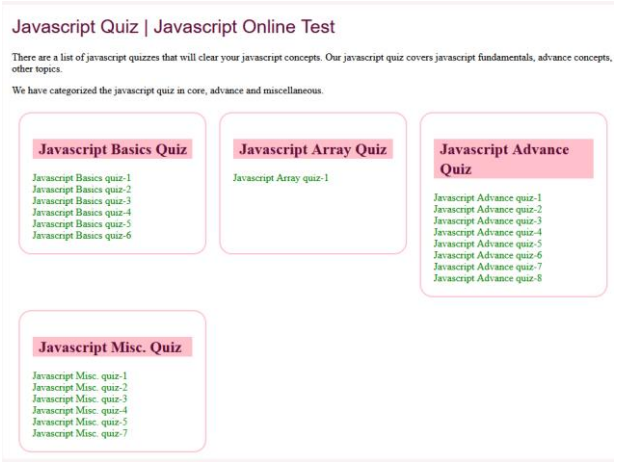
Game

https://js.checkio.org/

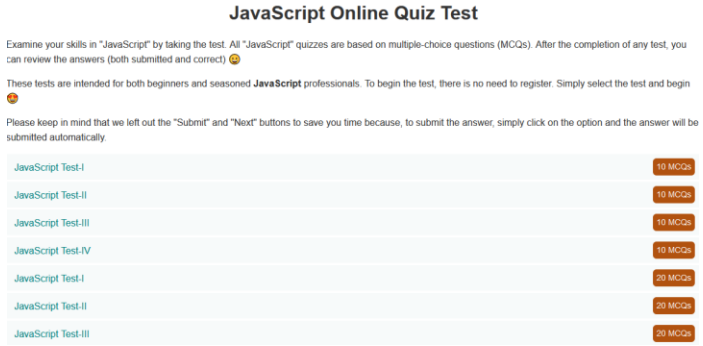
Quizzes



https://www.w3schools.com/quiztest/default.asp



https://www.javatpoint.com/javascript-quiz



https://codescracker.com/exam/showtest.php?subid=6

https://www.tutorialspoint.com/javascript/javascript_online_quiz.htm

źródło www.w3schools.com