zaprogramowani.com

szkoła programowania

WEBINAR NR 2

JAVASCRIPT

ALGORYTMY

CZYM SĄ ALGORYTMY?

Algorytm to:

- skończony ciąg czynności, który umożliwia przekształcić dane wejściowe na dane wyjściowe.

- opis rozwiązania danego problemu krok po kroku

ETAPY KONSTRUOWANIA ALGORYTMU

1. Ustalenie problemu do rozwiązania dla algorytmu
2. Określenie danych wejściowych
3. Określenie wyniku oraz sposobu w jaki zostanie zaprezentowany
4. Wybranie metody wykonania zadania, która wg nas jest najlepsza
5. Zapisanie algorytmu
6. Sprawdzenie poprawności rozwiązania
7. Przeprowadzenie testów dla różnych danych wejściowych
8. Praktyczna ocena skuteczności algorytmu

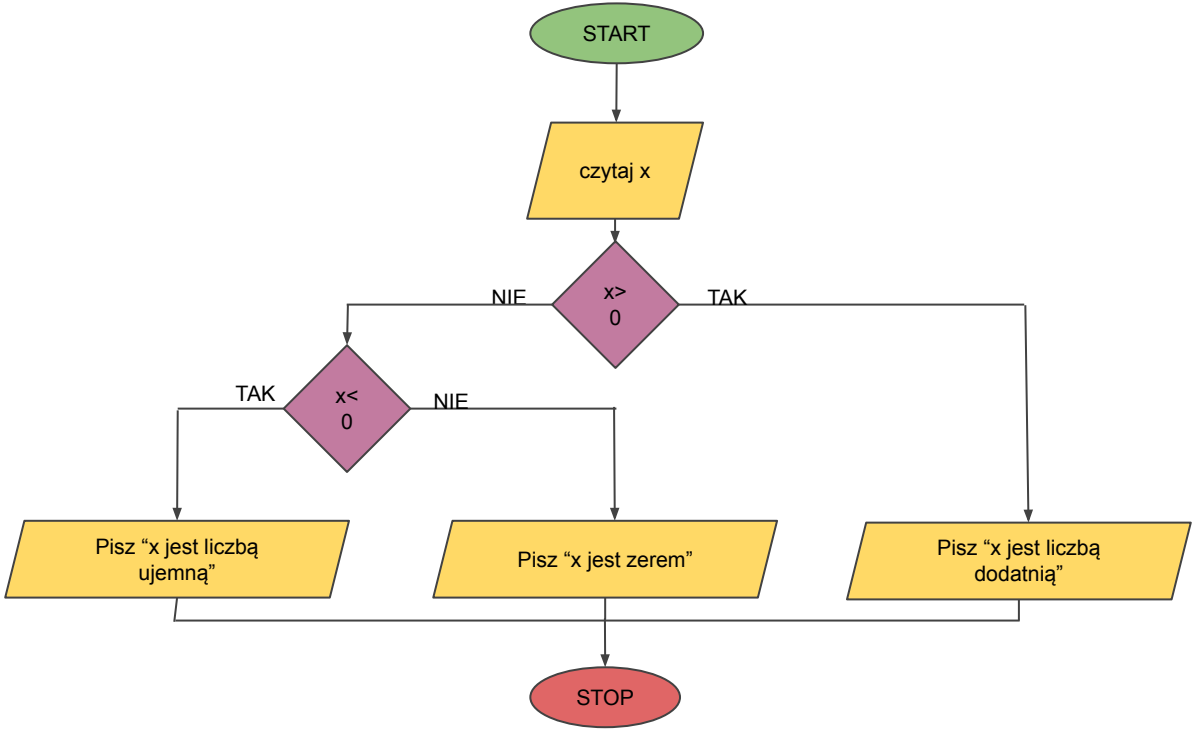
PRZYKŁADEM ALGORYTMU MOŻE BYĆ PRZEPIS KULINARNY, NP. NA KANAPKĘ! DANYMI WEJŚCIOWYMI SĄ SKŁADNIKI, WYKONYWANYMI OPERACJAMI SĄ WSZYSTKIE CZYNNOŚCI W TRAKCIE PRZYGOTOWYWANIA KANAPKI, A DANYMI WYJŚCIOWYMI JEST GOTOWA KANAPKA! :)

SPOSOBY ZAPISANIA ALGORYTMU

– – –

- opis słowny
- lista kroków
- pseudokod
- schemat blokowy - graficzna prezentacja algorytmu
- program komputerowy - zapis w danym języku programowania

SCHEMAT BLOKOWY



PROGRAM KOMPUTEROWY - ZAPIS W JAVIE

```
package javaapplication1;

public class JavaApplication1 {

    public static void main(String[] args){
        int x=7;

        if(x>0)
            System.out.println("X jest większe od 0.");
        else if(x==0)
            System.out.println("X jest równe 0.");
        else
                System.out.println("X jest mniejsze od 0.");

    }
}
```

RODZAJE ALGORYTMÓW

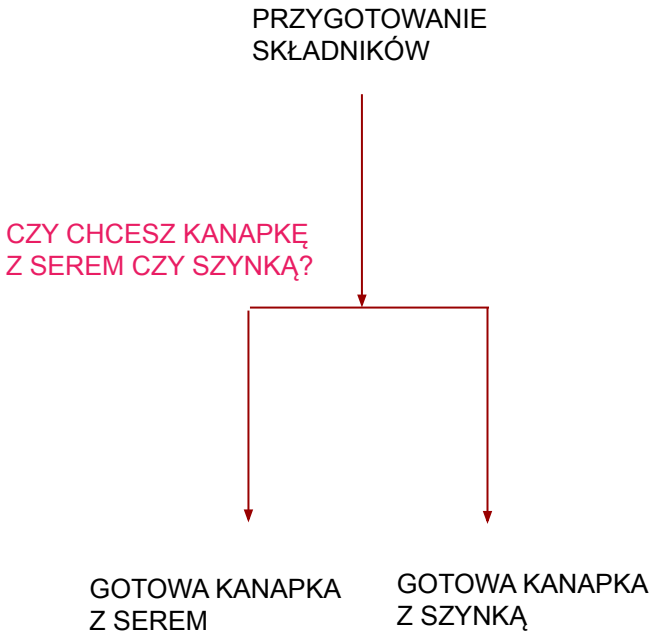
ALGORYTMY LINIOWE

To takie algorytmy, w których nie określono żadnych warunków, a każde z poleceń wykonywane jest bezpośrednio jedno po drugim. Takie algorytmy nazywamy również jako sekwencyjne.

JEST TO NAJPROSTRZY ALGORYTM. ROZPATRUJĄC PRZYKŁAD KANAPKI, NA POCZĄTKU PRZYGOTOWUJEMY SKŁADNIKI, NASTĘPNIE ROBIMY KANAPKĘ I NA KOŃCU JĄ JEMY ;)

ALGORYTMY WARUNKOWE

To algorytmy w których zostają postawione pewne warunki, i wykonanie konkretnych instrukcji jest uzależnione od ich spełnienia bądź nie.



ALGORYTM STAWIA JAKIŚ WARUNEK, PYTANIE KTÓRE W ZALEŻNOŚCI OD ODPOWIEDZI POPROWADZI NASTĘPNE DZIAŁANIA JEDNĄ Z DWÓCH MOŻLIWYCH ŚCIEŻEK.
W PRZYKŁADZIE KANAPKI, WARUNKIEM MOŻE BYĆ ZADECYDOWANIE CZY WOLIMY KANAPKĘ Z SZYNKĄ CZY Z SEREM.

ALGORYTMY ITERACYJNE

To algorytmy w których występuje powtarzanie pewnych operacji. Liczba powtórzeń może zostać z góry narzucona albo zależeć od spełnienia jakiegoś warunku, który za każdym powtórzeniem będzie sprawdzany.

Instrukcję powtarzania danego ciągu operacji nazywamy ITERACJĄ lub PĘTLĄ

POSTAWIONY WARUNEK, DOPÓKI NIE ZOSTANIE SPEŁNIONY, DOPÓTY NIE POZWOLI ABY WYKONANO KOLEJNE ZADANIA.
W PRZYKŁADZIE KANAPKI, DOPÓKI NIE MAMY MASŁA, TO NIE MOŻEMY PRZYGOTOWAĆ KANAPKI.

SCHEMATY BLOKOWE

CZYM SĄ SCHEMATY BLOKOWE?

SCHEMATY BLOKOWE

Schematy blokowe umożliwiają przedstawienie algorytmu w postaci symboli graficznych, które opisują wszystkie wykonywane operacje w programie wraz z ich kolejnością.

Pomagają zrozumieć problematykę danego zadania do rozwiązania.

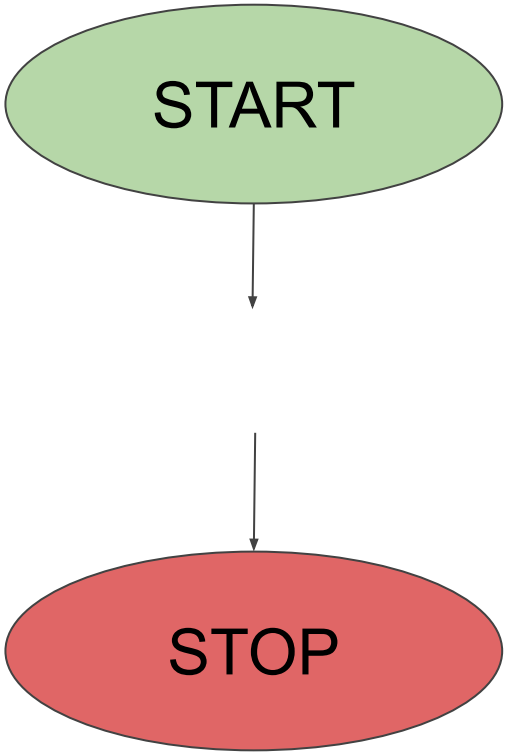
Pomagają w rozwiązaniu skomplikowanych zagadnień.

BLOKI GRANICZNE

Bloki te oznaczają początek oraz koniec danego algorytmu. Mają kształt owalu.

Każdy algorytm ma tylko jeden blok START, którego może wychodzić tylko jedno połączenie.

Każdy algorytm może posiadać kilka bloków STOP (co najmniej jeden).



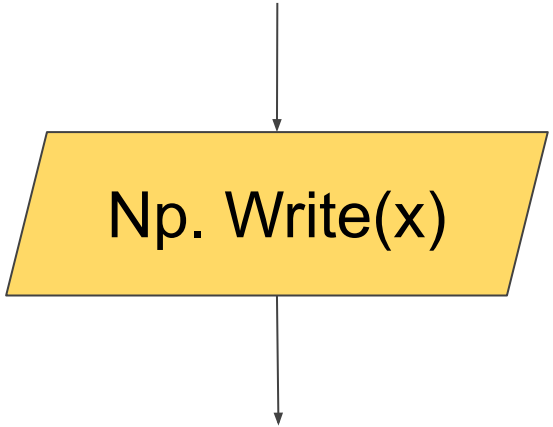
BLOK WEJŚCIA / WYJŚCIA

Blok ten odpowiada za wykonanie operacji wprowadzenia lub wyprowadzenia danych, wyników czy komunikatów.

Ma kształt podłużnego równoległoboku.

Może posiadać tylko jedno połączenie wejściowe oraz jedno połączenie wychodzące.

W jednym schemacie może znajdować się wiele takich bloków.



BLOK OPERACYJNY

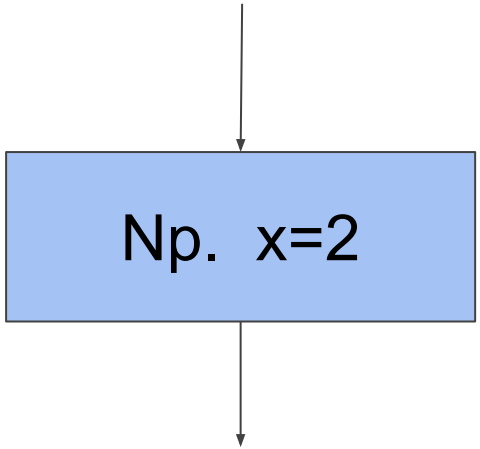
Blok odpowiedzialny za wszelkie operacje w programie, które zmieniają wartości zmiennych, np. obliczenia.

Bloki operacyjne posiadają kształt prostokąta.

Jeden blok może zawierać więcej niż jedną operację.

Może posiadać tylko jedno połączenie wejściowe oraz jedno wychodzące.

W jednym schemacie może znajdować się wiele takich bloków.



BLOK DECYZYJNY

Blok ten pozwala na wybranie jednej z dwóch możliwych ścieżek działania.

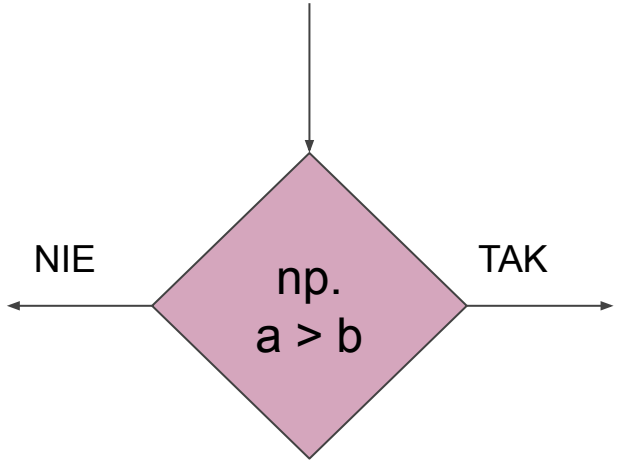
Ma kształt rombu.

Posiada tylko jedno wejście.

Posiada dwie ścieżki wychodzące:

TAK - jeśli wpisany warunek został spełniony

NIE - jeżeli wpisane warunek nie został spełniony

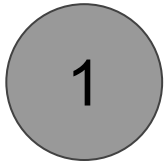


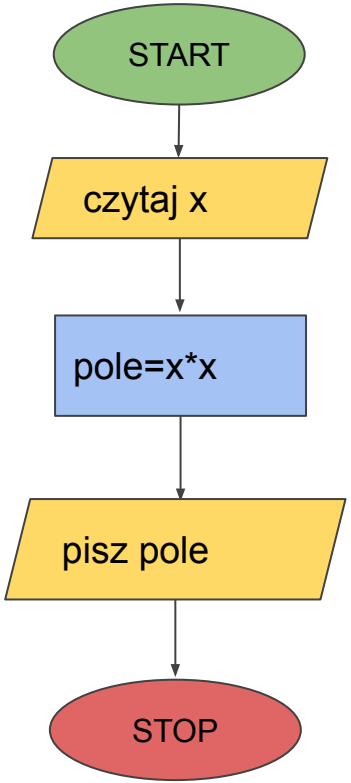
POŁĄCZENIE

Połączenie odpowiedzialne jest za łączenie ze sobą bloków w schemacie. Określa kierunek w którą stronę przebiega przepływ danych czy też kolejność wykonywanych działań. Może też się łączyć z innymi połączeniami w jedno.

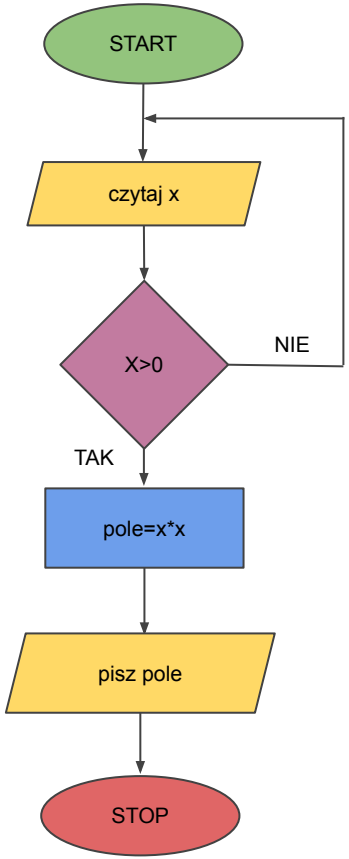
ŁĄCZNIK

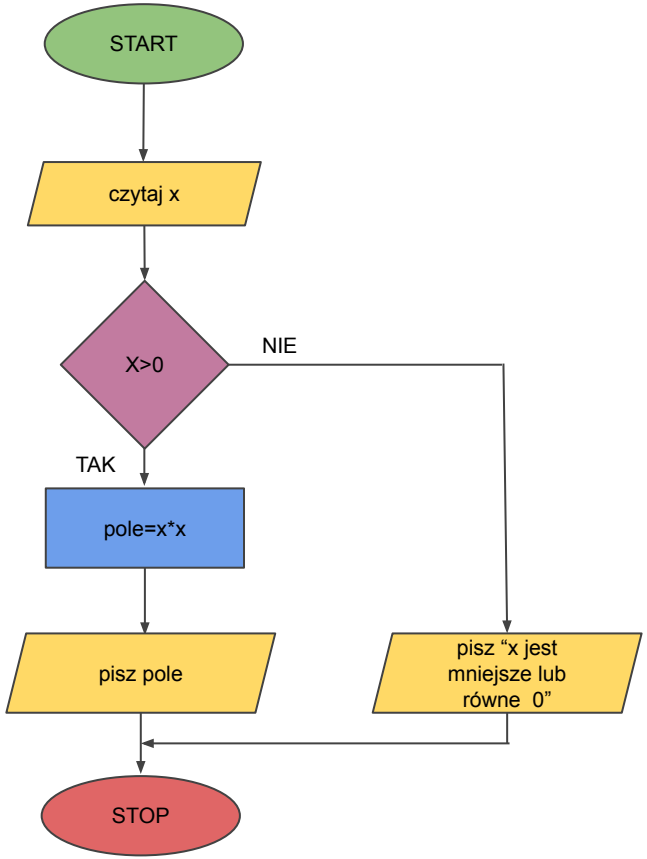
Stosujemy je, kiedy schemat blokowy chcemy narysować w kilku częściach. Jest to odsyłacz do innego fragmentu. Umieszczone oznaczenie wewnątrz musi być identyczne w obu częściach.





ALGORYTM NA OBLICZANIE POLA KWADRATU





ALGORYTM NA OBLICZANIE POLA KWADRATU, POD WARUNKIEM, ŻE X>0

JAVASCRIPT

CZYM JEST JavaScript?

Jest to skryptowy język programowania.

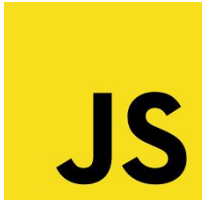
Wykorzystywany podczas pisania stron internetowych.

Pozwala na stworzenie interakcji pomiędzy użytkownikiem, a stroną internetową.

Java != JavaScript

Java

- Bardziej backend
- Różne aplikacje
- Do pisania potrzebujemy JDK
- Kod musi być skompilowany
- Kod jest ukryty po skompilowaniu
- Samodzielny język
- Język silnie typowany



JavaScript

- Bardziej frontend
- Głównie aplikacje webowe
- Do pisania wystarczy notatnik
- Kod od razu gotowy
- Kod jest dostępny dla każdego
- Do działania potrzebny HTML
- Język słabo typowany

ZASADY TWORZENIA

Aby utworzyć arkusz js i móc go używać należy:

1. Utworzyć nowy arkusz o rozszerzeniu **.js**
2. Arkusz należy “przypiąć” do pliku strony z rozszerzeniem html

Zrobimy to dodając odpowiednią linijkę do naszego kodu strony na samym końcu tuż przed znacznikiem </body>:

```
<script src="ścieżka pliku"></script>
```

DOŁĄCZANIE SKRYPTÓW WEWNĄTRZ KODU HTML

Podobnie jak **CSS**, skrypt możemy pisać bezpośrednio w pliku **HTML**, ale jest to niepoprawna praktyka.

Wystarczy cały kod pisać pomiędzy znacznikami **<script> </script>**

PODSTAWOWE FUNKCJE

Podstawowe Funkcje

alert(‘Hello World!’); - funkcja, która wywołuje okno alertu na stronie

console.log(‘Hello World!’); - funkcja, która drukuje w konsoli treść zawartą w argumencie

Podstawowe Funkcje

prompt('tekst'); - funkcja, która wywołuje okno z możliwością wprowadzenia treści
//można np zapytać użytkownika o imię, a następnie przypisać je do zmiennej

confirm('tekst'); - funkcja, która działa podobnie do **alert**, ale zwraca wartość logiczną, którą można przypisać do zmiennej i później wykorzystać

Te funkcje dziś nie są już stosowane

ZMIENNE

Zmienne w JavaScript

Zmienna to **“pudełko”**, które przechowuje wartości do niej przypisane.

Każda stworzona zmienna musi mieć swoją **unikatową nazwę**.

Zmienne w JavaScript

JavaScript to język słabo typowany.

Do zmiennej możemy przypisać **dowolny typ danych**, w zależności od potrzeb.

Mogą to być **liczby całkowite, zmiennoprzecinkowe, teksty, wartości logiczne.**

LET vs CONST vs VAR

Zmienne w JavaScript

Const - zmienna stała, nie można wartości później zmienić

Let - zmienna do których można przypisywać wielokrotnie różne wartości

Var - zmienna ogólna, dziś już nie używana ale spotykana

Typy danych

Do stworzonych zmiennych możemy przypisywać różne typy danych. Podstawowe to:

Number - wartość liczbowa

String - wartość znakowa

Boolean - wartość logiczna (prawda lub fałsz)

Undefined - wartość domyślna

Null - wartość bez wartości (typ wyświetli się jako objekt)

Inicjowanie zmiennej

```
typ nazwa = wartość;
```

Łączenie wyświetlenia tekstu i wartości zmiennej

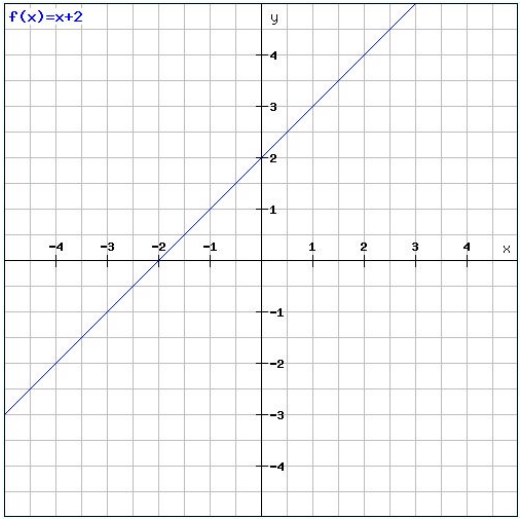
Możemy w łatwy sposób wyświetlać zarówno tekst jak i wartości przypisane do zmiennej jednocześnie przy pomocy pojedynczej funkcji.

```
const name = 'Adam';

let age = 20;

console.log(`Mam na imię ${name} i mam ${age} lat!`);
```

FUNKCJE

Funkcja w matematyce

$$f(x) = x+2$$



Funkcja w JavaScript

```
function add(x) {
    return x+2;
}
```

Funkcje

Funkcje służą do wykonywania zadanych działań na naszej stronie.

Funkcja pobiera argument, następnie wykonuje na nim operacje i zwraca wynik.

Występują również funkcję bez określonego argumentu oraz bez zwracania wartości.

Składnia:

```
function nazwa(argument){
        instrukcje;
        return wynik;
}
```

Wywołanie:

```
nazwa(argument);
```

Funkcje

Zwrot funkcji możemy zapamiętać przypisując wywołanie funkcji do zmiennej.

Umożliwia to późniejsze wykorzystanie wartości zwrotnej funkcji.

Przykład:

```
let a=2;
let b=3;

function calc(x,y){
        return x+y;
}
let suma = calc(a,b);

console.log(`suma wynosi ${suma});
```

Po co?

Zasada **DRY - don’t repeat yourself**

dzięki funkcji możemy uprościć nasz kod, deklarując w niej instrukcje, które będziemy wielokrotnie wykorzystywać

Większa czytelność kodu.

Łatwiej znaleźć i poprawić błąd.

INTERAKCJE

CZYM JEST DOM?

DOM (Document Object Model), obiektowy model dokumentu, to wygenerowana przez przeglądarkę strona internetowa, na podstawie instrukcji jaką jest napisany **kod HTML.**

JavaScript może modyfikować **DOM**.

zwracanie node'a

Aby móc dostać się do konkretnego elementu w naszym **DOM**u, musimy posłużyć się metodą **querySelector**, a w jej argumencie podać selektor, do którego się odwołujemy:

```
document.querySelector('.selektor')
```

metoda innerHTML

Aby móc wyświetlić cały selektor, wystarczy przypisać go do zmiennej, a następnie ją wyświetlić. Jeżeli chcemy wyświetlić jedynie zawartość tekstową takiego selektora posłużymy się metodą **innerHTML**:

```
let tresc = document.querySelector('.selektor')

console.log(tresc.innerHTML);
```

nadpisanie treści

Treść w selektorze możemy również nadpisywać, co będzie miało wpływ na naszą stronę internetową i treść na niej wyświetlaną.

```
tresc.innerHTML = 'Hello World!';
```

tworzenie przycisku

Aby utworzyć przycisk na naszej stronie, należy użyć odpowiedniego znacznika w kodzie HTML:

```
<button>tekst</button>
```

nasłuchiwanie zdarzeń

Do wykonania zdarzenia na danym elemencie musimy posłużyć się metodą **addEventListener** oraz określić typ zdarzenia oraz podpiąć funkcję je obsługującą.

```
element.addEventListener(*typ*, *funkcja*);
```

<button> != <a>

<button> - interakcja
<a> - nawigacja

KONIEC CZ. 2

Dzięki za udział ;)

ŹRÓDŁA OBRAZÓW:

STR. 5 “składniki kanapki” [źródło: https://www.kids-world.com/en/plantoys-play-food-sandwich-p-75280.html, dostęp 18/03/2020]
STR. 10 “kanapka” [źródło: https://www.empik.com/woodyland-zestaw-do-zabawy-dla-dzieci-kanapka-woodyland,p1141071645,zabawki-p, dostęp 18/03/2020]
STR. 24,26 “logo JavaScript” [źródło: https://en.wikipedia.org/wiki/Metropol_Parasol, dostęp 18/03/2020]
STR. 26 “logo Java” [źródło: https://www.iprsam.com/pl/2018/12/19/nie-ma-juz-bezplatnej-wersji-oracle-java-se/, dostęp, 18/03/2020]
STR. 47 “znak instrukcja” [źródło: https://www.pngitem.com/middle/ixoxxhb_file-iso-m-svg-consult-instructions-for-use/, dostęp, 18/03/2020]