JavaScript - wstęp

Czym jest JavaScript?

JavaScript to obiektowy język programowania, wykorzystywany bardzo często na stronach internetowych, aby sprawić, by były bardziej interaktywne i atrakcyjniejsze dla użytkownika.

Wszystkie obliczenia i operacje, zapisane w tym języku skryptowym, są wykonywane na tzw. front-endzie, czyli „po stronie użytkownika” – to znaczy, że to komputer osoby przeglądającej jest obciążany, gdy wykonywany jest program.

W przeciwieństwie do JS, język PHP jest językiem back-endowym, czyli to serwer, na którym umieszczona jest strona odpowiada za przeprowadzenie obliczeń dla programu.

Co możemy zrobić dzięki JS?

- dynamicznie zmieniać styl obiektów na stronie (np. dodawać i zabierać im klasy)
- podmieniać treść w blokach
- pobierać treść od użytkownika
- tworzyć elementy animowane
- pisać proste „minigry”
- itd.

Programy składają się z wielu funkcji

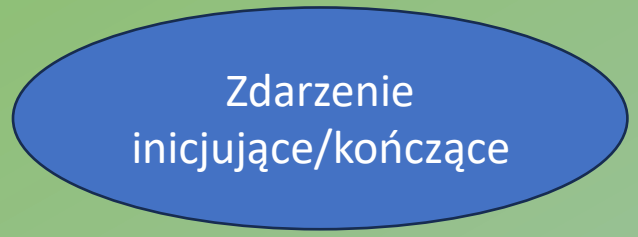
Program to kod, który ma wykonać określone zadanie, realizując polecenia zapisane jako jego kolejne linijki w określonym języku programowania.

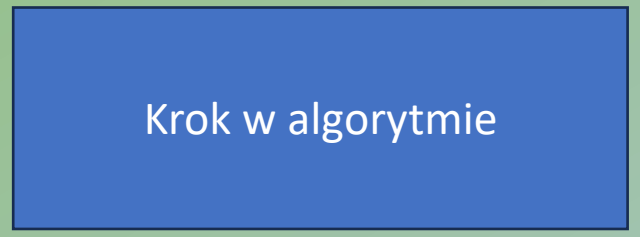
Funkcja to fragment programu, który może przyjmować pewne parametry (wartości) wejściowe i być wykorzystywany wiele razy w programie.

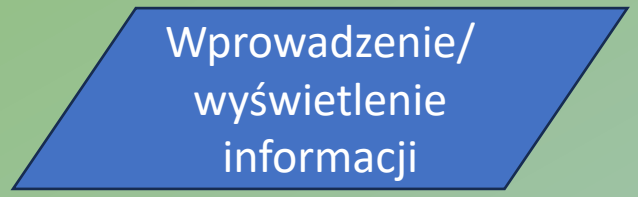
Podstawy algorytmiki

- Stałe i zmienne – fragmenty pamięci, w których przechowywana jest jakaś wartość, do której program ma dostęp i może ją wykorzystywać
- Tablice – zmienne wielowymiarowe, mogące przechowywać wiele wartości
- Warunki i opcje – program będzie bardzo często musiał podjąć jakąś decyzję: który krok wykonać w zależności od tego, jakie warunki zaszły w trakcie wykonywania skryptu (np. jaka jest wartość zmiennej)
- Pętle – niektóre operacje trzeba będzie wykonać kilka razy, zanim program się skończy. Zamiast pisać te operacje jedna pod drugą, można umieścić je w pętli, czyli takim kawałku kodu, który będzie wykonywał się raz po raz, dopóki nie zostanie spełniony warunek wyjścia

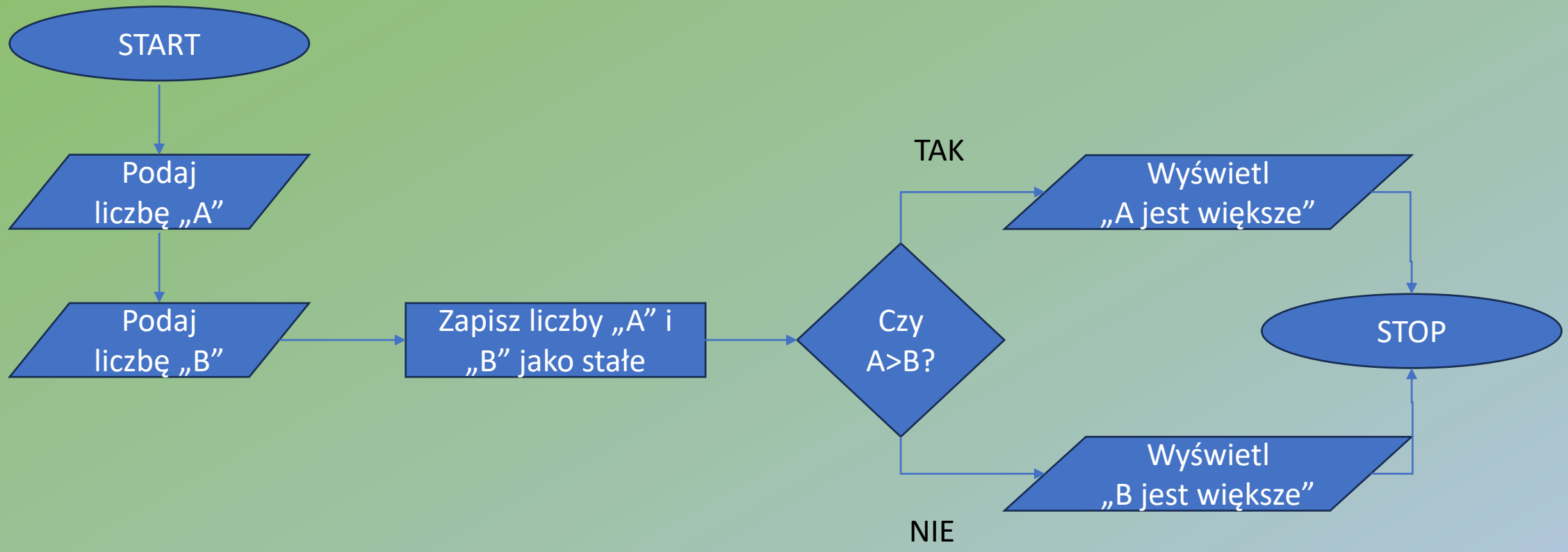
Symbole stosowane w algorytmice







Logika algorytmiczna – sprawdzenie, która liczba jest większa



Jak to przełożyć na język JavaScript?

- const – stała, niezmienna wartość
- var/let – zmienne wartości
- if – warunek decyzyjny
- /* komentarz */

```
/* deklaracja stałych */
const A;
const B;
/* przypisanie wartości do stałych */
A = 4;
B = 6;
/* sporządzenie warunku „czy A jest większe od B?” */
if (A > B) {
        /* w klamrach tworzy się nowy blok kodu */
        /* ten blok wykona się tylko, jeżeli warunek A > B jest prawdziwy */
        /* console.log(treść) wyświetla coś w konsoli przeglądarki */
        console.log(„A jest większe”);
}

else {
        /* ten blok kodu wykona się tylko, jeżeli warunek A > B jest fałszywy */
        console.log(„B jest większe”);

}
```

JavaScript samemu określa typ danych

Co będzie wynikiem polecenia:

```
const a = „1”;
const b = „1”;
const c = a + b;
console.log(c);
```

Co będzie wynikiem polecenia:

```
const a = 1;
const b = 1;
const c = a + b;
console.log(c);
```

Jak podłączyć skrypt do strony internetowej?

```
<script>
        treść kodu JavaScript
</script>
```

ALBO (lepsza metoda)

```
<script src=„folderZeSkryptami/skrypt1.js”>
```

Czym jest obiekt?

Obiekt to pewien fragment rzeczywistości, posiadający swoje cechy, przełożony na język, zrozumiały dla komputera.

W JavaScript (i innych językach obiektowych) obiekt posiada atrybuty, które z kolei posiadają wartości.

Przykładowym obiektem może być „telewizor”. Jego atrybutami mogą być: marka, model, przekątna (w calach), czy jest „Smart”. Wartościami tych atrybutów mogą być: „Samsung”, „Smart TV 1000”, „70”, „tak”.

Obiekt: telewizor

Atrybut (właściwość)	Wartość
Marka	Samsung
Model	Smart TV 1000
Przekątna (w calach)	70
Czy jest „smart”?	tak

Wszystkie dane przechowywane w środowisku komputerowym mają swój „typ” który mówi, jakiego rodzaju jest to informacja. Dane mogą być typu:

- liczbowego:
  - całkowitoliczbowego, czyli np. 1, 2, 3, 4, itd, ale nie 3.14,
  - zmiennoprzecinkowego, czyli np. 3.14, 21.37,
- tekstowego (np. „Samsung”),
- Bool’owego (prawda/fałsz, tak/nie, 1/0),
- itd.

Wszystko na stronach internetowych jest obiektem – model DOM

DOM – Document Object Model – sposób przedstawienia dokumentów HTML jako model obiektowy – byt składający się z wielu obiektów powiązanych ze sobą w „węzły” (nodes).

Struktura strony internetowej może być porównana z drzewem, z którego rozchodzą się gałęzie, na których rosną mniejsze gałęzie… aż do pojedynczych liści. Każda gałąź i liść (obiekty) mają swoje cechy (atrybuty).

Skąd skrypt ma wiedzieć, skąd ma brać dane?

Każdy element (znacznik) strony internetowej ma pewne atrybuty, np.:

Obiekt: div
Atrybuty: id, class
Ich wartości: „bloczek1”, „bloczkiGorne”

JavaScript może wykorzystać te atrybuty, by odnaleźć konkretny jej fragment i pobrać z niego wartości tychże atrybutów. Innymi słowy, każdy element strony jest „obiektem”.

Jak pobrać wartość atrybutu danego obiektu?

HTML:

```
<div id=„bloczek1”>
        Hello World!
</div>
```

JS:

```
const blokZtekstem = document.getElementById(„bloczek1”);
const tekst = blokZtekstem.innerText;
console.log(tekst)
```

Słowa kluczowe w JS

- let – nowa zmienna
- var – nowa zmienna
- const – nowa stała
- if ( ) – warunek „jeżeli”
- for ( ) – pętla
- = - przypisanie
- ==, >=, <=, >, <, != - porównania
- switch ( ) case – „inny” warunek „jeżeli”

Z czego może składać się fragment kodu JS?

```
let klocek;
klocek = document.getElementByID(„blok”);
klocek.innerText = „Nowa treść w klocku”;
klocek.classList.add = („nowaKlasa”);
klocek.classList.remove = („staraKlasa”);
```

polecenie, obiekt/zmienna, metoda, wartość

Metoda to pewna funkcja, która ma zostać wykonana na danym obiekcie.

A co z tą zmianą stylu?

Tylko JS:

```
let klocek = document.getElementById(„klocek”);
klocek.style.color = „red”;
klocek.style.margin = „10px”;
klocek.style.border = „3px dashed blue”;
```

CSS + JS:

CSS:

```
.nowaKlasa {
    color:red;
    margin: 10px;
    border: 3px dashed blue;
}
```

JS:

```
let klocek = document.getElementById(„klocek”);
klocek.classList.add(nowaKlasa);
```

Ważniejsze polecenia w JS

- document.getElementById(identyfikator) – szuka obiektu o id = identyfikator
- document.getElementByClassName(klasa) – szuka obiektów o klasie = klasa
- document.getElementByTagName(znacznik) – szuka obiektów będących danym znacznikiem
- obiekt.innerText – zawartość tekstowa danego obiektu
- obiekt.innerHTML – zawartość hipertekstowa danego obiektu
- console.log(treść) – wyświetlenie treści w konsoli przeglądarki

- document.querySelector(„p.podpis”) – znajduje pierwszy z brzegu (patrząc od góry strony) element o typie <p> i klasie podpis
- document.querySelector(„li.testowa.naNiebiesko”) – znajduje pierwszy z brzegu element o typie <li> i klasach „testowa” oraz „naNiebiesko”

Do jednego klocka można przypisać wiele klas w ten sposób: <li class = „testowa naNiebiesko”>

- document.querySelectorAll(„article”) – znajduje wszystkie elementy typu <article> i przechowuje je w tzw. liście węzłów (node list) do której można się odwoływać jak do tablicy

Tablice

Tablica to zmienna o kilku wymiarach.

Zmienna let x = 2 ma tylko jeden wymiar i, co ważniejsze, jedną wartość

Zmienna let tablica[] również ma jeden wymiar, ale może przechowywać wiele wartości np.:

```
let owoce[];
owoce.push(„banan”, „truskawka”, „jabłko”);
console.log(owoce[1]);
```

Co będzie rezultatem?

Tablice + pętle = <3

Dane w tablicach można „załadować” od razu przy jej tworzeniu, np.:

```
let tablica2 = [„Alpha”, „Beta”, „Charlie”, „Delta”];
for (i = 0; i < tablica2.length; i++) {
        console.log(tablica2[i]);
}
```

Co będzie rezultatem?

Wielowymiarowe tablice?

```
let tablica3 [] [];
tablica3[0][0] = „Wes Anderson”;
tablica3[0][1] = „Wyspa Psów”;
tablica3[1][0] = „Christopher Nolan”;
tablica3[1][1] = „Mroczny Rycerz”;
for (i = 0; i <= 1; i++) {
        for (j = 0; j <=1; j++ {
                console.log(tablica3[i][j]);
        }
}
```

Co będzie rezultatem?