JS Core Concepts

Agenda

JS Types and Variables

JS Control Flow

JS Higer-Order and Pure Functions

JS Data Structures

JS Input / Output

DOM

console

”prompt, confirm, alert” functions

Types and Variables

```
let name = "John";

var grade = 5;

const isStudent = true;
```

Variable scope

String Formatting

```
let wiek = 21;

let cv1 = "Piotr ma " + wiek + " lat";

let cv2 = `Piotr ma ${wiek} lat`;
```

Conditional Execution

```
if (warunek) {
  instrukcje
} else {
  instrukcje
}
```

```
switch (expression) {
  case value1:
    // Statements
    [break;]
  ...
  case valueN:
    //Statements
    [break;]

  [default:
    //Statements
    [break;]]
}
```

Loops

```
while (warunek) {
  instrukcje
}
```

```
for(let n=0; n<=N; n++)
{
  instrukcje
}
```

Defining a Function

```
const sayHi = function() {
  console.log("Hello everyone!");
}

sayHi();
```

Function declaration

```
function sayHi() {
    console.log("Hello everyone!");
}

sayHi();
```

Arrow Functions

```
const sayHi = () => {
    console.log("Hello everyone!");
}

sayHi();
```

Arrow Functions Details

[Arrow Functions (short syntax) - JavaScript Tutorial (ES6)]

Higher-Order Functions

```
let total = 0;
for (let n=1; n<=10; n++) {
  total += i;
}
console.log(total);
```

```
console.log(sum(range(1,10));
```

Function as a argument

Return a function

Pure Functions

Return value the same for the same arguments

No side effects

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

Functional Programming

Pure Functions

Learning Functional Programming with JavaScript

Arrays

```
shoppingList = ['pizza','juice'];

console.log(shoppingList);

shoppingList.push('chips');

console.log(shoppingList);

console.log(shoppingList.length);
```

Array Loops

```
for (let i=0; i<shoppingList.length; i++) {
    console.log(shoppingList[i]);
}

for (let item of shoppingList) {
    console.log(item);
}

shoppingList.forEach(item => {console.log(item);});
```

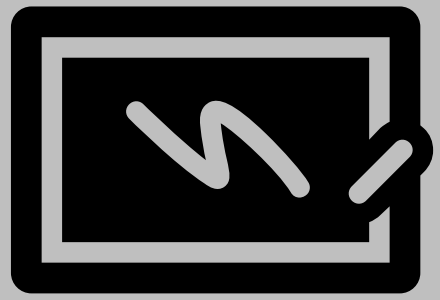
Objects

```
class Student {
  constructor(name, university) {
    this.name = name;
    this.university = university;
  }
  sayHi() {
    console.log(
      `I'm ${this.name}, studying at ${this.university}!`
    );
  }
}

student = new Student('Ania', 'UEK’);
student.sayHi();
```

Maps

```
let Students = {
    "Ania"   : "312009",
    "Wojtek" : "312011"
}

console.log(Students);
console.log(Students["Wojtek"]);
Students["Monika"] = "312224";
console.log(Students);
```

To do



Strings

Program address.js zawiera dane adresowe.

Wyświetl dwukrotnie dane adresowe na konsoli w formie podanej w programie:

nie stosując formatowania łańcuchów znakowych

stosując formatowanie łańcuchów znakowych

String Formatting

Program heron.js zawiera długości boków trójkąta.

Korzystając z wzoru Herona, oblicz pole trójkąta. Dla wyznaczenia pierwiastka kwadratowego, użyj Math.sqrt().

Wyświetl rezultat stosując formatowanie łańcucha znakowego.

Loops

Program loop.js wyświetla zbiór wartości naturalnych.

Zmodyfikuj program, aby wyświetlał te wartości z użyciem instrukcji while (loopwhile.js) oraz instrukcji for (loopfor.js).

Pure Functions

Firma posiada trzy pojazdy. Samochód może być sprawy (true) lub niezdatny do jazdy (false).

W pliku cars.js uzupełnij funkcję isCar(car1, car2, car3), która powinna zwracać prawdę, jeśli wszystkie samochody są sprawę lub jeśli wszystkie samochody są niesprawne. W pozostałych przypadkach funkcja powinna zwracać fałsz.

Arrow Functions

Utwórz program trapezium.js, który obliczy i wyświetli na konsoli pole powierzchni trapezu o wymiarach a, b oraz h.

Jako argument console.log zastosuj funkcję strzałkową, która dla argumentów a, b oraz h zwraca pole powierzchni trapezu.

Functions Set

Program ordering.js zawiera funkcję odwracającą kolejność elementów tablicy. Zwróć uwagę na składowe tej funkcji.

Uzupełnij funkcję zwracającą posortowane elementy tablicy.

Uzupełnij funkcję zwracającą parzyste elementy tablicy.

Sprawdź działanie utworzonych funkcji.

Math Object Methods

Utwórz program random.js, w którym:

wypełnij tablicę dziesięcioma liczbami naturalnymi z przedziału <5,20>

wyświetl na konsoli zawartość tablicy

Do wykonania obydwu powyższych operacji użyj funkcji strzałkowych.

Classes

Utwórz program person.js z klasą Person. Obiekt klasy Person przechowuje podstawowe dane osobowe, tj. imię oraz nazwisko przekazane w momencie tworzenia obiektu.

Rozszerz klasę o metodę, która:

zwraca imię i nazwisko w formacie ”Imie NAZWISKO”

zwraca inicjały osoby w formacie ”I.N.”

Na podstawie klasy Person utwórz dwa obiekty: jeden reprezentujący Jana Nowaka, a drugi reprezentujący Ciebie. Sprawdź działanie metod (wyświetl ich rezultaty na konsoli).

Object methods

W skład programu university.js wchodzi klasa Students. Obiekt tej klasy przechowuje listę studentów (ich imion) oraz umożliwia przetwarzanie tej listy.

Uzupełnij wszystkie metody zawarte w klasie, aby możliwe było wyświetlenie informacji na konsoli zgodnie z zawartymi w programie instrukcjami.

Sprawdź, czy wszystkie informacje wyświetlane na konsoli są poprawne.