W JS aby wyświetlić wartość variable używa się nastepujacej funkcji:

* Console.log()

Aby stworzyć variable stosuje się:

* Let variable = ...

Możemy stworzyć również variable, którego wartość nie będzie mogła ulec zmianie, stosując Const.

* Const PI = 3,14

Jesli chcemy zrobić komentarz na wiele lini to stosuje się:

/\* tutaj wpisz tekst \*/

Aby sprawdzić typ variable można to zrobić przy pomocy typeof:

Console.log(typeof variable)

Istotna strona gdzie możemy sprawdzić kolejność egzekowwania operatorów:

Mdn operator precedense >

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_precedence>

Aby dodać do anszego string dynamicznie wartości wykorzystujemy do tego ``:

Let test = 8

Console.log(`This is ${test}`)

Dodatkowo tworzenie stringów przy pomocy `` pozwala na łatw tworzenie nowych linijek tekstu.

`test

test`

Warunek if else, stynatx: if (warunek typu coś > od czegoś) { baza, która zadziała jesli warunek zostął spełniony} else { }

Istnieje również else if block aka elif z pythona:

If (michal) {

} else if (ada) {

} else {}

Jeśli warunek if ma tylko jedną część to nie musimy tworzyć {} np.:

If (age === 18) console.log(`This is true`)

Jeśli chcemy manualnie zmienić typ wartości aka type conversion:

Aby uzyskać string: String(x)

Aby uzyskać liczbę: Number(x)

Javascript automatycznie zamienia wartości aka type cohersion np:

`23` - `10` - 3 da nam liczbę 10, ale jesli zrobimy: `23` + `10` + 3 to dostaniemy string 2313

Truthy and Falsy values

Istnieje 5 nieprawdziwych wartości: 0, ‘’, undefined, null, NaN (not a number).

W przypadku gdybyśmy chcieli sprawdzić przy pomocy warunku if czy variable jest True, a posiadałby jedną z powyższych wartości to dsotalibyśmy False.

**Equality Operators == i ===.**

Potrójny === nazywa się strict equality operator i spełniony ejst tylko jeśli obywdwie przyrównywane wartości sa dokładnie takie same.

Podwójny == nazywa się lose equality operator i np gdy przyrwónujemy „18” == 18 dostaniemy True. Zasadniczo staramy się unikać tego.

Uzyskanie wartości od użytkownika aka input, w tym celu używa się funkcji **prompt**:

const user\_value = prompt(„what’s your favorite color”)

Boolean operators = and, or, not(!) < tak jak w Python.

And operator jest zapisywany jako &&,

Or operator jest zapisywany jako ||,

Not operator jest zapisany jako !n,

**Switch statments:**

Działają na zasadnie bloków if ale z inną kosntrukcją np.:

Switch(day) {

case ‘monday’:

consolo.log(„…”)

consolo.log(„…”)

**break**

case ‘tuseday’:

consol.log(„…”)

**break**

case ‘wednesday:

case ‘thursday: (zadziałą dla obydwu rzeczy)

consol.log(„…”)

**break (jest potrzebny bo inaczej kod przejdzie do dalszej częsci)**

default:

consol.log(“Not a valid day”)

}

**Statements and Expressions**

Expressions to np. 3 +4, 1991, true && false && !false

Statements to większa część kodu np. if else statements.

**Conditional (Ternary) Operator**

Poniżej jest budowa conditional operatora, gdzie tworzymy warunek, następnie znakiem ? tworzymy blok spełnionego założenia i po : blok negatywnego spełnienia. Pozwala to zaoszczędzić dużo czasu przy prostych sprawdzeniach warunku if else.

Const age = 24

age >= 18 ? console.log(“I like alcohol”) : console.log(„I like wataa”)

Z przypisaniem do variable:

Const drink = age >= 18 ? “wine”: “water”

**Strict Mode**

Celem strict mode jest umożliwienie włączenie widzenia błędów oraz uniemożliwia wykonanie pewnych funkcji.

Aby aktywować strict mode należy dodać na początku skryptów **„use strict”**

**Funkcje**

Funkcje tak jak w python, to blok kodu, który można w łatwy sposób ponownie wykorzystać. Konstrukcja:

Function Blob(parameter1, parameter2) {

Console.log(parameter1, parameter2)

}

**Functions Declarations vs Expressions**

Obydwie formy pisania funkcji są właściwe, osobiście preferuje f. declaration. Różnica między nimi polega, że declaration f. może być wezwana przed jej blokiem.

Function declaraction:

function calcAge1 (birthYear) {

return 2033 - birthYear

}

const age1 = calcAge1(1991)

console.log(age1)

Function expression:

const calcAge2 = **function** (birthYear) {

return 2037 - birthYear

}

const age2 = calcAge2(1991)

console.log(age1)

**Arrow Function**

Jest to forma funkcji o bardzo prostej budowie, nadająca się do prostych funkcji, która automatycznie zwraca wartość swojego ciała.

Const calcAge3 = birthYear => 2033 – birthyear

Console.log(calcAge3(1990))

A co w przypadku bardziej rozbudowanej funkcji:

Const yearsUntilRetirement = birthYear => {

const age = 2037 – birthYear

const retirement = 65 - age

return retirement

}

W przypadku wielu parametrów:

Const yearsUntilRetirement = (birthyear, firstName) => {

const age = 2037 – birthYear

const retirement = 65 - age

return `${firstName} retirement in ${ retirement }`

}

**Functions Calling Other Functions**

Dla mnie logiczne.

Const cutFrutPieces = function (fruit) {

return fruit \* 4

}

Function fruitProcessor(apples, oranges) {

Const applePieces = cutFrutPieces(apples)

Const orangePieces = cutFrutPieces(oranges)

const juice = `Juice with ${applePieces} apples and ${orangePieces} oranges`

return juice

}

re