**JavaScript**

JavaScript yra aukšto levelio, OOP, multi paradigminė programavimo kalba. Multi-paradigm dėl to, kad JS galima naudoti įvairius programavimo stilius. Norint sukurti gražius, interaktyvius ir dinaminius tinklapius naudojam HTML,CSS,JS.

HTML – atsakingas už tinklapio turinį. Tekstas, nuotraukos, mygtukai ir visas turinys kurį matome tinklapyje.

CSS – atsakingas už turinio pristatymą, trumpai tariant stiliui ir išdėstymui.

JS – leidžia pridėti dinaminius ir interaktyvius efektus betkokiam tinklapiui.

**Data Types**

Duomenų tipai gali būt primityvai arba objektai. Primityvas yra tik tokiu atveju, jeigu jis nėra objektas. Yra 7 primityvūs tipai: „number“, „String“, boolean, „undefined“, „null“, „symbol“ ir „big int“.

* Number – tas, pats kas Float. Jis visada turi skaičių po kablelio, bet jeigu tai INT – jo nesimato.
* String – žodis/sakinys sudarytas iš raidžių. Turi būti parašyta kabutėse (““, ‘’)
* Boolean – loginis tipas, jis nusako arba true arba false.
* Undefined – kai kintamajam nėra priskiriama jokia vertė (let children;)
* Null – neturi tokios vertės.
* Symbol – vertė yra unikali ir negali būti pakeista.
* BigInt – Gali laikyti didesnį skaičių kiekį negu Number tipas.

JavaScript turi „dynamic typing“ – kuris reiškia, kad sukūrus naują kintamąjį, mes neturime nurodyti jo duomenų tipo. Tai leidžia kintamąjį su tuo pačių vardu (pavyzdžiui x) iš pradžių nurodyti kaip „Number“ o vėliau, kode, tą patį kintamąjį (x) pakeist į String ar kokį kitą.

**let, const, var**

* **let** - geriausia naudoti, kai žinom, kad kintamojo vertė gali keistis. Arba jeigu pradžioj paskelbiam „undefined“ kintamąjį ir vėlesniam kode jį keisim.
* **const** – naudojam tada, kai paskelbtas kintamasis ir jo vertė ateityje niekada nesikeis. Tai galėtų būti gimimo diena. Trumpai tariant- kintamasis, kuris nemutuos. Const negali neturėti vertės, vertė jam visada privalo būti priskirta.
* **var** – senasis būdas kaip paskelbiamas kintamasis ir jis veikia maždaug taip pat kaip let.

JavaScript leidžia mums paskelbti kintamąjį be let/const/var ir jis sėkmingai veiks, bet tai yra blogas būdas, nes tuomet kintamasis nepapuls į mums norimą scope ir bus paskelbtas kaip globalus kintamasis.

**Basic Operators**

* Mathematical/arithmetic operators ( + | - | / | \* )
* Assignment operators ( = | += | \*= | /= | -= | ++ | --)
* Comparison operators ( > | < | >= | <= )

**Conversion/Coercion**

**Type conversion** yra tuomet kai manualiniu būdų konvertuoji iš vieno tipo į kitą. Type coercion yra kai JS automatiškai konvertuoja tipus (už scenos) vietoj mūsų. Konvertuoti galima String į Number, jeigu String’e yra nurodyti skaičiai, jeigu nurodytos raidės ar žodis, gaunamas NaN – Not a Number eroras. Galima konvertuoti iš Number į String. Ir galim konvertuoti į boolean.

**Strict mode**

Strict mode – yra specialus rėžimas, kuris gali būti aktyvuojamas JavaScript‘e, kuris leidžia mums lengviau rašyti saugų JS kodą. Kad jį aktyvuoti tereikia pradžioje skritpo parašyti ‘use strict‘; Taip pat šį modą galima aktyvuoti tik kažkokiai konkrečiai funkcijai ar blokui. Žodis saugiau, šiuo atveju, reiškia, kad developeriams leidžia lengviau išvengti klaidų. Šis rėžimas leidžia atrasti bugus kode dėl dviejų priežasčių:

1. Draudžia mums daryti kai kuriuos dalykus.
2. Sukuria matomas klaidas tam tikrose situacijose, kuriose be šio modo JS paprasčiausiai failintų be leidimo žinoti, kad mes padarėme klaidą.

**Functions**

Funckijos gali gauti duomenis, transformuoti ir išleisti (output). Iš viso yra trys funkcijų tipai:

* Function Declaration – gali būti panaudojamos prieš paskelbiant funkciją. Tai yra kodo viršuje.
* Function Expression – iš esmės funkcijos vertę patalpina į kintamąjį.
* Arrow – gerai naudoti greitom, vienos linijos, funkcijoms. Bet neturi this žodžio.

Kiekviena funkcija turi turėti pavadinimą, parametrus, kurie yra nurodomi skliausteliuose (), turi turėti kūna (aprašyta visa funkcijos veikla) ir dažniausiai turi return teiginį. Taip pat betkuri **funkcija kuri yra pridedama prie objekto vadinama metodu.**