Методы и свойства у переменных примитивного типа в JS

При выводе методов у примитива JS сама создает объект от примитива тем самым давая доступ к методам

Компиляторы и интерпретаторы

Исходный код 🡪 компилятор 🡪 Бинарный код 🡪 Движок КОМПИЛЯТОР

Исходный код 🡪 Движок ИНТЕРПРЕТАТОР

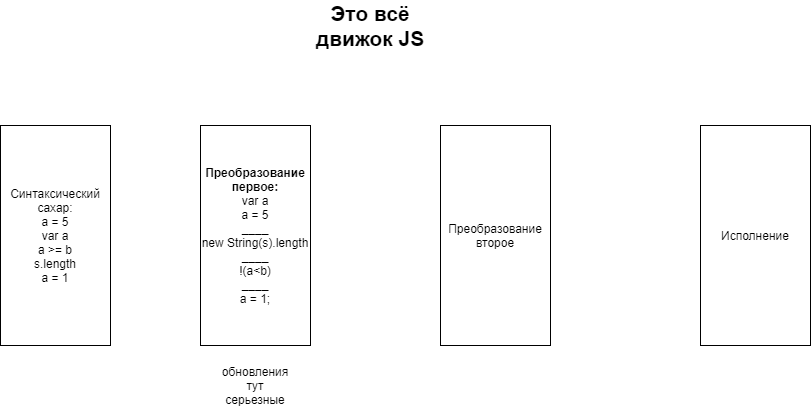
Во втором случае движок есть интерпретатор

Третий случай – компиляция в момент выполнения

Исходный код 🡪 Движок(содержит в себе компилятор и компилирует внутри движка)

Результатом работы движка является процесс

Компиляция кода в JS



Оператор Typeof в JS

Строка - String, число - number, логический тип - boolean, большие числа - bigInt, Null - Object, Неопределенность - undefined, Symbol - Symbol, Object – Object

Исключением является объект типа function – function

Тип Symbol в JS

Использовали в функции для создания уникального айди чтобы защитить объекты от хака

Два способа обращения к свойствам объекта в JS(. [])

var o = {};

o.qqq = 1;

o["kkk"] = 2;

c(o,o["qqq"],o.kkk); // есть отличие:

o["hello world"] = true; // разделитель пробелом

c(o["hello world"]);

// другое отличие в том что для внесения переменной в названия свойств - надо использовать скобки

Циклы часть 2

var a = ["e1","e2","e3"];

for (var i in a){ // индексы

    print(i);

}

for (var i of a){ // значения

    print(i);

}

var o = {a:"p1",b:"p2",c:"p3"};

for (var i in o){ // перебор названий свойств

    print(i);

}

for (var i of o){ // ошибка (нет перебора значений)

    print(i);

}

Можно сделать массив свойств и пройти по нему:

for (var i of Object.values(o)){

    print(i);

}

GETTER SETTER

var module = {};

module.\_\_defineGetter\_\_("version",

    function() {

        print("Version getter");

        return this.\_\_version;

    }

)

module.\_\_defineSetter\_\_("version",

    function(x) {

        print("Version setter");

        this.\_\_version = x;

    }

)

module.version = "1.1";

print(module.version);

Методы defineSetter,defineGetter,lookUpSetter,lookUpGetter в JS

Метод Object.defineProperty

Метод Object.create

var moduleSuperNew = Object.create(module, // модуль - прототип для нового модуля(\_\_proto)

    {

        dependencies:

        {

            configurable: false,

            enumerable: false,

            set: function(x){

                this.\_\_dependencies = x;

            },

            get: function(){

                return this.\_\_dependencies;

            }

        },

        name:

        {

            configurable: false,

            enumerable: false,

            get: function() {

                return "SUPER NEW MODULE";

            }

        }

    });