**Objects**

Стректуры который может в себе хранить любые типы данных в формате

ключ: значение

Так же могут быть вложеными.

Примеры и синтаксис:

**#1 обращение к свойству**

"use strict";

const options = {

    name: 'test',

    width: 1024,

    height: 1024,

    colors: {

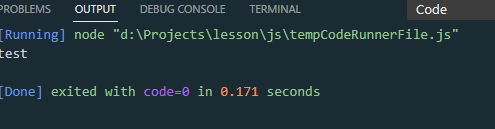
        border: 'black',

        bg: 'red'

    }

};

console.log(options.name);



**#2 свойство удаление какой-то опции**

"use strict";

const options = {

    name: 'test',

    width: 1024,

    height: 1024,

    colors: {

        border: 'black',

        bg: 'red'

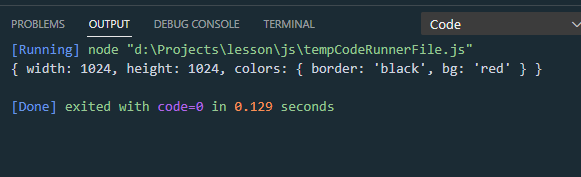
    }

};

//console.log(options.name);

delete options.name;

console.log(options);



**#3 перебор всех свойств объекта (for in)**

По стандарту каждое свойство называется ключом (key)

"use strict";

const options = {

    name: 'test',

    width: 1024,

    height: 1024,

    colors: {

        border: 'black',

        bg: 'red'

    }

};

//console.log(options.name);

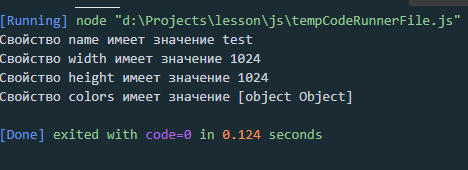
//delete options.name;

//console.log(options);

for (let key in options) {

    console.log(`Свойство ${key} имеет значение ${options[key]}`);

}



**Последнее свойство джава скрипт не может вывести так как это не строка а объект**.

Как результат вывод происходит последнее значение (строковое представление обьекта). Это можно исправить задав условия.

"use strict";

const options = {

    name: 'test',

    width: 1024,

    height: 1024,

    colors: {

        border: 'black',

        bg: 'red'

    }

};

for (let key in options) {

    if (typeof(options[key]) === 'object') {

        for (let i in options[key]) {

            console.log(`Свойство ${i} имеет значение ${options[key][i]}`);

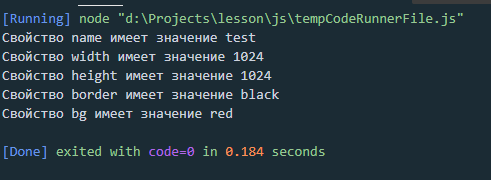
        }

    } else {

        console.log(`Свойство ${key} имеет значение ${options[key]}`);

    }

}



**Свойства и методы**

Прием **счётчика (сколько свойств было внутри)**

**counter**

"use strict";

const options = {

    name: 'test',

    width: 1024,

    height: 1024,

    colors: {

        border: 'black',

        bg: 'red'

    }

};

let counter = 0;

for (let key in options) {

/\*    if (typeof(options[key]) === 'object') {

        for (let i in options[key]) {

            console.log(`Свойство ${i} имеет значение ${options[key][i]}`);

            counter++;

        }

    } else {

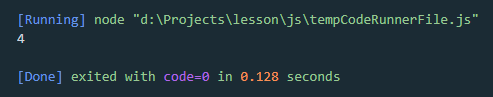
        console.log(`Свойство ${key} имеет значение ${options[key]}`);

        \*/counter++;

//    }

}

console.log(counter);



**#1 Object.keys (получить все свойства с ключами) + length (количество)**

"use strict";

const options = {

    name: 'test',

    width: 1024,

    height: 1024,

    colors: {

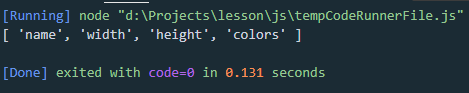
        border: 'black',

        bg: 'red'

    }

};

console.log(Object.keys(options));



"use strict";

const options = {

    name: 'test',

    width: 1024,

    height: 1024,

    colors: {

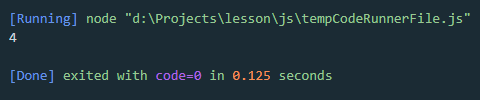
        border: 'black',

        bg: 'red'

    }

};

console.log(Object.keys(options).length);



**#2 Методы так же можно создавать вручную.**

Например:

"use strict";

const options = {

    name: 'test',

    width: 1024,

    height: 1024,

    colors: {

        border: 'black',

        bg: 'red'

    },

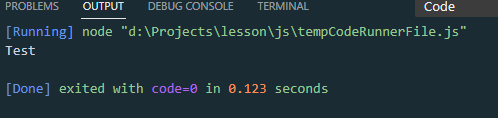
    makeTest: function() {

        console.log("Test");

    }

};

options.makeTest();



**#3 Могут иметь встроенные свойства:**

Например:

makeTest: function() {

        console.log("Test");

    }

};

options.makeTest();

**#4 Деструктизация обьекта** (стандарт Еs6) для того что бы достучаться к нужным свойствам у обьекта

Вытаскивание свойств в качестве отдельных структур.

Примеры:

"use strict";

const options = {

    name: 'test',

    width: 1024,

    height: 1024,

    colors: {

        border: 'black',

        bg: 'red'

    },

    makeTest: function() {

        console.log("Test");

    }

};

options.makeTest();

const {border, bg} = options.colors;

console.log(border);

