**Вопросы 💎**

1. Чем отличается класс от объекта? Приведите примеры и того, и другого, отличные от указанных в уроке.

An object is a set of properties and/or functions gathered within a single variable. Each property is a name-value pair. A class is a template that describes the properties and methods that any object based on that class will have. A class is just a structure, while objects are specific variables with fields filled in based on that structure.

Example: class Reptile & object Alligator; class WebsiteVisitor & object MobileUser; class Women & object SingleWomen

1. Приведите примеры объектно-ориентированного подхода в реальной жизни.

*Возможный вариант ответа: например, библиотека могла бы содержать классы Книга (название, автор, год выпуска, количество страниц), Сотрудники (ФИО, должность, зарплата, дата начала работы) и Читатели (ФИО, контакты, номер читательского билета).*

*An example here can be:*

*a Gym can include following classes Device (name, purchase year, number of programs, target muscles), Visitor (name, age, paymenttype, begin of contract), Employee (name, age, work years);*

*Сomputer (operational syste (installation year, type, memory), menu (apps, anti-virus, office), graphic objects (), devices (mouse, keyboard, speakers))*

Приведите примеры объектно-ориентированного подхода в программировании.

*Возможный вариант ответа: например, товары в интернет магазине (название товара, тип, цена), комментарии в соцсети (автор комментария, дата, текст комментария), видео в Youtube (название видео, его адрес, автор и дата загрузки).*

*Online banking (login, password, name, transactions, address of account owner);*

*e-mail (login, name, memory)*

1. Что такое конструктор? Самостоятельно изучите и напишите, какие бывают виды конструкторов.

A constructor function describes a template by which objects (instances) will be created. This template is based on an enumeration of the properties and methods that the object, created from a single structure (class), will have.

**Built-In Constructors**

new String() // A new String object

new Number() // A new Number object

new Boolean() // A new Boolean object

new Object() // A new Object object

new Array() // A new Array object

new RegExp() // A new RegExp object

new Function() // A new Function object

new Date() // A new Date object

**User-defined Constructor**

This can be any constructor the user wants to write, for example:

function Human(firstName, lastName) {

this.firstName = firstName

this.lastName = lastName

}

Custom constructors must always contain the keyword "this". This allows you to store and access the unique values of the instance being created. The instances are created with the "new" keyword.

const liza = new Human('Liza', 'Monetochka')

console.log(liza.firstName) // Liza

console.log(liza.lastName) // Monetochka

**Adding a property to an object**

Adding a new property to an existing object is simple:

myFather.nationality = "English";

The property will be added to the myFather object, but not to the myMother object. (Or any other person type object).

**Adding a method to an object**

Adding a new method to an existing object is simple:

myFather.name = function () {

return this.firstName + " + this.lastName;

};

The method will be added to the myFather object, but not to the myMother object. (Or any other object of type person).

**Adding a property to the object constructor**

You cannot add a new property to the object constructor in the same way as you would add it to an existing object.

To add a new property to the constructor, you have to add it to the constructor function:

function Person(first, last, age, eyecolor) {

this.firstName = first;

this.lastName = last;

this.age = age;

this.eyeColor = eyecolor;

this.nationality = "English";

}

The object properties can be set to default values.

**Adding a Method to Object constructor**

A constructor function can also define methods:

function Person(first, last, age, eyecolor) {

this.firstName = first;

this.lastName = last;

this.age = age;

this.eyeColor = eyecolor;

this.name = function() {return this.firstName + " + this.lastName;}

}

You can't add new methods to an object constructor in the same way as you do to an existing object. Adding methods to the object must be done inside the constructor function:

function Person(firstName, lastName, age, eyeColor) {

this.firstName = firstName;

this.lastName = lastName;

this.age = age;

this.eyeColor = eyeColor;

this.changeName = function (name) {

this.lastName = name;

};

}

The changeName() function assigns the value of the name parameter to the lastName property of the person object:

myMother.changeName("Doe");

JavaScript knows which object we're talking about by "substituting" the myMother object in the this keyword.

1. Что выведет код? Почему именно так?

Console will show ninja as a result of a function bike(), because the bike function was defined. However, it will show an error in the next two cases obj1.bike() and obj2.bike(), because the variables obj1 and obj2 are just separate variables, which are not linked to the function bike.

1. Чем статические свойства и методы отличаются от нестатических? В каких ситуациях они применяются?

Class methods and properties that are not bound to a specific object and can exist without creating an instance and the word new are called static. Usually static methods are used to implement functions belonging to a class, but not to any specific object of that class.

For example, there are `Article` objects, and we need a function to compare them. The natural solution is to make an `Article.compare` method to do this.

It is very convenient to use static methods to create calculator classes, utility classes, helper classes to work with some other classes.

Usually static methods are used to implement functions belonging to a class, but not to any specific object of that class.

1. Самостоятельно изучите, что такое геттеры и сеттеры, приведите пример класса с их использованием.

Getters and setters are methods whose task is to control access to fields. A getter reads and returns the value of a field, while a setter takes the value as an argument and writes it to the field. This makes it possible to equip such methods with additional handlers. For example, when writing a value to an object field, the setter can check the type or whether the value is within the valid range (validation). You can add lazy initialisation or caching to the getter, if the actual value actually lies in the database. There are many possible applications.

let obj = {

get propName() {

// getter, the code executed on getting obj.propName

},

set propName(value) {

// setter, the code executed on setting obj.propName = value

}

};

The getter works when obj.propName is read, the setter – when it is assigned.

For instance, we have a user object with name and surname:

let user = {

name: "John",

surname: "Smith"

};

Now we want to add a fullName property, that should be "John Smith". Of course, we don’t want to copy-paste existing information, so we can implement it as an accessor:

let user = {

name: "John",

surname: "Smith",

get fullName() {

return `${this.name} ${this.surname}`;

}

};

alert(user.fullName); // John Smith

From the outside, an accessor property looks like a regular one. That’s the idea of accessor properties. We don’t call user.fullName as a function, we read it normally: the getter runs behind the scenes.

As of now, fullName has only a getter. If we attempt to assign user.fullName=, there will be an error:

let user = {

get fullName() {

return `...`;

}

};

user.fullName = "Test"; // Error (property has only a getter)

Let’s fix it by adding a setter for user.fullName:

let user = {

name: "John",

surname: "Smith",

get fullName() {

return `${this.name} ${this.surname}`;

},

set fullName(value) {

[this.name, this.surname] = value.split(" ");

}

};

// set fullName is executed with the given value.

user.fullName = "Alice Cooper";

alert(user.name); // Alice

alert(user.surname); // Cooper

As the result, we have a “virtual” property fullName. It is readable and writable.

1. Что выведет код? Почему именно так?

Console will show John, because this is a value of a property name, which is assigned to a variable member.

1. Что выведет код? Почему именно так?

The console will show the following information: name: “Valera”, age: 23, city: “Amsterdam”, because we added a new property city to the object person.