Presentation on the topic “Inheritance in JavaScript. ES6 classes.”

Hello, my name is Andrew and today we will talk about inheritance and classes in the programming language JavaScript.

To understand what inheritance is, first of all we should mention the definitions of a class and an object.

In JavaScript, an object is an unordered dynamic collection of properties (key - value pairs).

Each key is a unique string. The value can be either a primitive, or a function or an object.

In JavaScript, everything except primitives is objects: objects, functions, and arrays.

We can create an object in several ways:

* object literals:
* functions;
* ES6 classes.

A class is some kind of schema or template with which you can create objects.

In JavaScript, a class is a function for creating a set of objects - ‘real’ instances of this class.

The new standard ES6 adds support for classes using the **class** keyword, followed by the class name. The body of the class is in braces.

A constructor is declared in the class body using the **constructor** keyword. A constructor is a special method that is called when an object is created, that is, when the **new** keyword is used. It creates the properties of the object and initializes it.

Properties are characteristics of an object. They describe which object or what it has. Properties are specified in the constructor. Each class member has these properties. In order to create properties for an object in the constructor, the **this** keyword is used, which indicates the object we are creating.

The constructor can take **parameters** that we can specify as **arguments** when creating a new object.

In class, we can create methods. Methods are what an object can do.

To specify a method, you must specify the **name of this method**.

In the body of a class, using the **static** keyword, you can declare static methods — that is, methods that belong to the class itself, and not to objects created on its basis.

In the body of the class, using the **get** and **set** keywords, we can add special properties — functions that look and behave like methods inside a class, and look and behave like properties outside.

Creating objects based on classes is one side of object-oriented programming. The other side is inheritance, that is, the ability of objects to inherit properties and methods from other classes.

In programming, inheritance can be divided into:

* classic - it is implemented in C, C #, JAVA and other strongly typed object-oriented languages. A feature of such inheritance is that classes are inherited from classes.
* prototype - used in JavaScript, LUA and other languages. The main difference from the classical is that in such inheritance objects are inherited from objects.

JavaScript inheritance has gone through several stages of development:

* crockford version
* ES5
* ES6

Crockford described inheritance using a multi-line function. And all of it was used for a long time before the advent of ES5.

In ES5, the **Object.create** method has been added and the function has been slightly modified.

Adding class support in ES6 has not introduced a new inheritance model. Inside, all the same prototypes are used. The syntax of classes in JavaScript is the so-called “syntactic sugar”. It simplifies working with classes and inheritance.

To define inheritance between two classes, the parent and the child, the **extends** keyword is used.

It is placed after the name of the child class and before the name of the parent class.

If the child class does not have a constructor, then it will use the parent constructor.

When using the parent constructor, the properties of the parent class are inherited.

In addition to inheriting properties, classes can also inherit methods.

If the child class has its own constructor, then this constructor should call the parent class constructor using the super keyword. This call must go before any properties are assigned.

In addition to methods, the child class can inherit static methods.

Also, a child class can inherit the getters and setters of the parent class.

That's all I wanted to tell you about inheritance and classes in the Javascript language.

Thanks.