OOP in JavaScript vs other languages

This lesson introduces OOP in JavaScript and compares it to other languages by going over the advantages/disadvantages of both.

WE'LL COVER THE FOLLOWING

JavaScript has relied on the object-based style of programming since the beginning as functions and methods can be found written in this style.

However, OOP in JavaScript works differently from other languages. If you're familiar with OOP in other languages, it's important that you put that knowledge aside for now, since holding on to those concepts might confuse you.

Now, let's look into how OOP in JavaScript differs from other languages.

OOP in Other languages

You must have seen that other languages such as C++, Java, and C# use the keyword `class` to define a class. A class has properties and methods in it for every instance of that class.

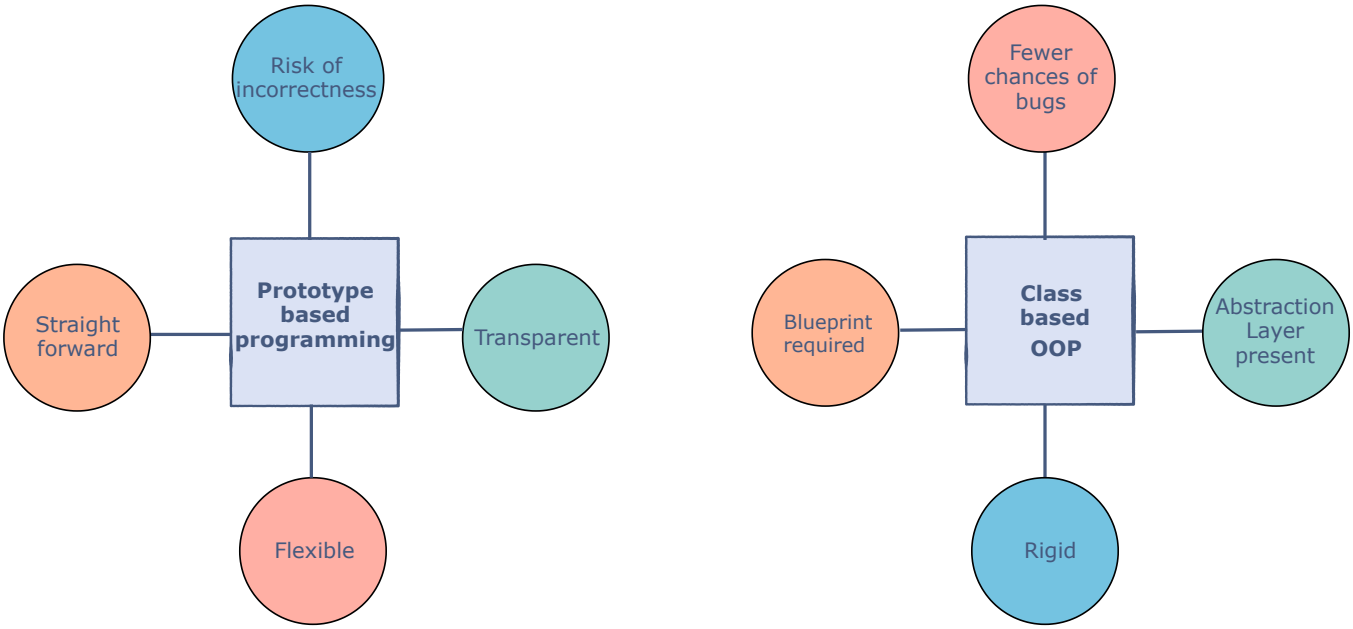
> **Note:** If you try to remove the `class` *keyword* the code will not compile and will generate an error.

The *class*, in this case, acts as a blueprint for the *object*. Or, you can say that the *object* will be an instance of this *class*.

OOP in JavaScript

However, we can also implement OOP without using classes, which is what JavaScript does. Before (and even after) introducing its ES2015 version, JavaScript relied on **Prototype-based programming**. In this programming style, the *object* encapsulates the *properties*, i.e., its *methods* and *data*, instead of a *class*. You can add new properties to this object whenever you feel like. So, an *object* can be an individual instead of an instance of the class; meaning, if you want an *object*, you can easily create one without having to create a class first.

Which Is Better?



Prototype-based vs Class-based programming

Conclusion

Both *Prototype-based* and *Class-based* OOP have their advantages and disadvantages.

Prototype-based is more straightforward as you don't need to create a blueprint beforehand. A blueprint requires planning in advance for properties that will be needed before creating an object, but it's possible that you might not be able to predict everything that will be needed while making the blueprint.

You can create the object directly since a class is not required and, due to complete transparency, its working can be observed easily. This also offers

flexibility. Hence, any changes to the objects can easily and quickly be made while they're being used.

Despite all these advantages in Prototype-based programming, there is also a higher risk of incorrectness as abrupt changes can be made just as easily. Since blueprints lay out a plan beforehand in a Class-based approach, there are fewer chances of bugs arising.

In the end, both approaches have their pros and cons, so the choice depends on both the programmer as well as the type of problem being dealt with.

Now that you know a little bit about OOP in JavaScript let's discuss the two versions of it, ES5 and ES6, in the next lesson.