Netcracker

Front-end development course

JavaScript

Lecture #6

More on OOP in JS

OOP in general

With Object-oriented approach we describe everything in object terms - they are being created, they change their properties, interact with other objects. Object terms come naturally from real world: “User”, “Person”, “Menu” and so on. In OOP most of object should represent familiar entity with methods and data.

Class in OOP - is a “template” for creating object of specific type.

Object-oriented analysis and design, by Grady Booch and co.

Design Patterns, by Gang of Four

OOP in general

Three concepts that defines OOP:

- Inheritance
- Encapsulation
- Polymorphism

SOLID - are advanced principles of building object-oriented architecture.

S - Single responsibility principle - God objects are prohibited.

O - Open/Closed principle - Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification Bertrand Meyer.

L - Liskov Substitution Principle - Derived classes behaviour must not contradict base class behaviour.

I - Interface Segregation Principle - More small interfaces is better.

D - Dependency Inversion Principle - We should depend from abstractions and not from specific implementation (good article on that topic).

OOP in JS

In JavaScript OOP and Classes can be organized in many different way:

- In classic prototype style
- In functional style
- In new ES6 “classes” style

Most common way until ES6 is “prototype style”.

Class.prototype - makes sense only for constructor functions, it’s only purpose is to set __proto__ for new objects.

By default every function has a prototype property, and object created from that function has a reference to their constructor.

Classes in JS

New syntax for classes came out with ES6. Under the hood classes inheritance still works the same way (with prototype chain).

Although, there are some differences between classes and function constructors:

- classes can’t be used without “new” - there will be an error.
- class declaration behave itself like “let” - with block scope

Classes can have special method “constructor” that will be invoked during creation with “new”.

Classes can have: getters and setters, static methods, parent methods and parent constructor can be accessed with “super”

Modules in JS

Module in JS - is just a file.

Most common approaches to modules are CommonJS (implemented in node.js) and ES6 modules (from ECMA specification)

Good article about ES6 modules

Q&A

Thank You

Netcracker