








Frontender[1.0] JavaScript - КЛАССЫ - БАЗА, поля, методы, get & set, #private, static. Часть 1

 YouTube	https://youtu.be/P5msG9OjXtY
 Telegram	https://t.me/Dmitry_Kolotilshikov
 Github	https://github.com/DmitryKolotilshikov/
 Boosty	https://boosty.to/dmitry_ko
# Номер урока	42

Классы в JavaScript. Часть 1

 Полезные ссылки: <https://learn.javascript.ru/class>

 В объектно-ориентированном программировании **класс** – это **расширяемый шаблон кода для создания объектов**, который устанавливает в них начальные значения (свойства) и реализацию поведения (методы).

Часто нам надо создавать много объектов одного вида, например пользователей, товары или что-то ещё. С этим может помочь функция-конструктор, мы ее знаем по прошлому уроку о прототипах. Но, начиная с ES6, есть классы.

Классы в JavaScript — это удобный способ работы с объектами и наследованием, который делает код более структурированным и читаемым. Они отлично подходят для построения сложных приложений и реализации объектно-ориентированного подхода. Они были добавлены в стандарте ECMAScript 2015 (ES6)

В JavaScript класс – это **разновидность функции**.



Создание класса

Классы создаются с помощью ключевого слова `class`. Внутри класса можно определять:

- **Конструктор** (`constructor`) — метод, вызываемый при создании объекта.
- **Методы** — функции, которые работают с данными объекта.
- **Статические методы** — функции, доступные на самом классе, но не на экземплярах.



Геттеры (`get`) и сеттеры (`set`) позволяют управлять доступом к свойствам объекта.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <!-- <script src="script.js"></script> -->
  <script src="shop.js"></script>
  <title>Classes</title>
</head>
<body>
</body>
</html>
```

```
// script.js
// Classes - Классы

const log = console.log;

// ----- Базовый синтаксис -----
/*
  class MyClass {
    // методы класса
    constructor() { ... }
    method1() { ... }
    method2() { ... }
    method3() { ... }
    ...
  }

  - Вызов new MyClass() создает новый объект со всеми перечисленными методами;
  - При этом автоматически вызывается метод constructor(), в нём можно инициализировать объект;
*/

class User {
  constructor(name) {
    this.name = name;
  }

  greeting() {
    log(`Hello, I am ${this.name}`)
  }
}
```

```

    greet = () => {
        log(`Hello, I am ${this.name}`)
    }
}

const userAlex = new User("Alex");
userAlex.greeting();

const userOlga = new User("Olga");
userOlga.greet();

// В JavaScript класс – это разновидность функции
log(typeof User); // function

// ----- new Function - Функция-конструктор -----

function UserFnClass (name) {
    this.name = name;

    this.greeting = function () {
        log(`Hello, I am ${this.name}`)
    }
}

UserFnClass.prototype.greet = function () {
    log(`Hello, I am ${this.name}`)
}

/*
    со стрелочной функцией тут потеря контекста (this)

    UserFnClass.prototype.greet = () => {
        log(`Hello, I am ${this.name}`)
    }
*/

const userFnVlad = new UserFnClass("Vlad");

log(userFnVlad.name);
userFnVlad.greeting();
userFnVlad.greet();

// ----- Геттеры и Сеттеры. Класс с приватными полями -----

class User1 {
    profession = "software engineer";
    #skills = "";

    constructor(name) {
        this.name = name;
    }

    get skills() {
        return this.#skills;
    }

    set skills(newSkills) {
        if (typeof newSkills !== "string") return;
    }
}

```

```

        this.#skills = newSkills;
    }
}

const userAnna = new User1("Anna");

log(userAnna.profession);

userAnna.profession = "developer"
log(userAnna.profession);

log(userAnna.skills);

userAnna.skills = "html, css, js, ts, react";
userAnna.skills = 111;
log(userAnna.skills);

// ----- Класс со стическими методами и свойствами -----

class User2 {
    static #instanceCount = 0;

    constructor(name) {
        this.name = name;
        User2.#instanceCount++;
    }

    greeting() {
        log(`Hello, I am ${this.name}`)
    }

    static getInstanceCount() {
        return log(User2.#instanceCount);
    }
}

const userKen = new User2("Ken");
userKen.greeting();

const userNatalia = new User2("Natalia");
userNatalia.greeting();

User2.getInstanceCount();

```