Frontender[1.0] JavaScript - this, контекст, объект перед точкой. Call, Bind, Apply

YouTube	https://youtu.be/vIpnWnzEVic
Telegram	https://t.me/Dmitry_Kolotilshikov
Github	https://github.com/DmitryKolotilshikov/
Boosty	https://boosty.to/dmitry_ko
# Номер урока	40

💡 Полезные ссылки:

https://learn.javascript.ru/global-object

https://learn.javascript.ru/call-apply-decorators

https://learn.javascript.ru/bind

https://learn.javascript.ru/object-methods

💡 В **JavaScript** ключевое слово **this** используется для ссылки на текущий объект, в контексте которого выполняется код.
Другими словами
this - это объект перед точкой.
Контекст
this определяется не только местоположением, но и способом вызова функции.

💡

1. Глобальный контекст

В глобальной области видимости (`this` вне функции):

- В браузере `this` будет ссылаться на объект `window`.
- В Node.js `this` ссылается на объект `global`.

2. В методах объекта

Когда `this` используется в методе объекта, оно указывает на сам объект.

3. Контекст `this` в стрелочных функциях

Стрелочные функции (`=>`) ведут себя по-другому: они не имеют собственного `this`. Вместо этого `this` внутри стрелочной функции берется из окружающего контекста.

4. Контекст `this` в событиях (DOM)

Когда `this` используется в обработчике события, он ссылается на элемент, который вызвал событие.

5. `this` с `call`, `apply` и `bind`

Методы `call`, `apply` и `bind` позволяют вручную задавать, каким будет значение `this` внутри функции.

- `call` – вызывает функцию с определённым `this` и принимает аргументы по отдельности.
- `apply` – вызывает функцию с определённым `this`, аргументы передаются массивом.
- `bind` – возвращает новую функцию с определённым `this`, которую можно вызвать позже.

6. Потеря контекста `this`

Когда передаёте метод объекта в другую функцию, `this` может "потеряться" и не будет больше указывать на исходный объект. Это одна из распространённых ошибок в JavaScript.

```html
<!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>
    <title>This</title>
</head>

<body>
    <button id="myButton">Click me</button>
    <script>
        const button = document.getElementById('myButton');
        button.addEventListener('click', function () {
            console.log(this); // Выведет сам элемент button
        });
    </script>
</body>

</html>
```

```
// this

// 1) Глобальный контекст
console.log(this); // В браузере выведет window, в Node.js - глобальный объект

// ---------------------------------------------------------------------
// 2) Методы объекта
const person = {
  name: "Alex",
  sayHello: function() {
    console.log(`Hello, ${this.name}`);
  }
};

person.sayHello(); // Выведет "Hello, Alex"

// ---------------------------------------------------------------------
// 3) стрелочные функции и this
const person1 = {
  age: 33,
  sayAge: () => {
    console.log(`Age: ${this.age}`);
  },
  sayAgeWithContext: function() {
      console.log(`Age: ${this.age}`);
  }
};

person1.sayAge(); // Выведет "Age: undefined"
person1.sayAgeWithContext(); // Выведет "Age: 33"

// 4) см в index.html
// ---------------------------------------------------------------------
// 5) call, apply и bind
function greet (greeting, emoji) { // function, не стрелочная функция
    console.log(`${greeting}, ${this.name} - ${emoji}`);
}

const person = { name: "Alex" };

greet.call(person, "Hello", "💥"); // Выведет "Hello, Alex - 💥"
greet.apply(person, ["Hi", "💎"]); // Выведет "Hi, Alex - 💎"

const boundGreet = greet.bind(person);
boundGreet("Hey", "🟢"); // Выведет "Hey, Alex - 🟢"

// ---------------------------------------------------------------------
// 6) потеря контекста
const sayAge = person1.sayAgeWithContext;
sayAge(); // Выведет "Age: undefined"

const boundSayAge = person1.sayAgeWithContext.bind(person1);
boundSayAge(); // Выведет "Age: 33"
```