

Язык JavaScript (основы: контекст вызова)

A large, bold, black logo consisting of the letters 'J' and 'S' in a stylized, rounded font. The 'J' has a thick vertical stem and a curved bottom. The 'S' is also thick and rounded, with a large loop. The logo is centered on a bright yellow background.

КОНТЕКСТ ВЫЗОВА

Под **контекстом функции** понимается **объект**, в рамках которого выполняется функция. Обратиться к этому объекту внутри функции можно с помощью ключевого слова **this**.

Если функция не является методом объекта, то **this** будет указывать на **window** (глобальный объект).

```
function myFunc() {  
    return this;  
}
```

```
console.log(myFunc()); // window
```

!!!Не работает при "use strict" будет возвращён
undefined

МЕТОД ОБЪЕКТА

Если функция – это метод объекта, то `this` внутри тела функции будет указывать на него.

```
let myObj = {  
  propA: 27,  
  methA: function() {  
    return this.propA; // this = myObj  
  }  
};  
console.log(myObj.methA()); // 27
```

ПЕРЕДАЧА КОНТЕКСТА

В JavaScript можно явно указать контекст, в котором необходимо вызвать функцию.

Первый способ - это использовать метод call.

Синтаксис:

/ func - функция*

**context - контекст, в котором нужно вызвать функцию*

** arg1, arg2, ... - список аргументов, которые нужно передать функции */*

func.call(context[, arg1[, arg2[, ...]]]);

ПЕРЕДАЧА КОНТЕКСТА (ПРИМЕР)

// объект user

```
let user = {  
  name: 'Василий',  
  age: 27  
}
```

// объявление функции getUserAge

```
function getUserAge() {  
  return this.age;  
}
```

// вызов функции в контексте объекта user

```
console.log(getUserAge.call(user)); // 27
```

ПЕРЕДАЧА КОНТЕКСТА

Второй способ - это использовать метод `apply`. Данный метод аналогичен `call`. Единственное отличие `apply` от `call` заключается в том, что аргументы в нём указываются посредством массива.

Синтаксис:

`/* func - функция`

`* context - контекст, в котором нужно вызвать функцию`

`* argArray - массив аргументов, которые нужно
передать функции */`

`func.apply(context[, argArray]);`

Пример:

`let arr = [1,2,3];`

`console.log(Math.min.apply(null, arr)); //1`

ПРИВЯЗКА КОНТЕКСТА

Метод **bind** предназначен для явной привязки контекста (`this`) к функции. Он в отличие от методов `call` и `apply` не вызывает функцию. Механизм этого метода заключается в создании функции-обёртки, которая будет устанавливать необходимый контекст целевой функции.

Синтаксис:

/ func - функция*

** context - контекст, с которым нужно связать функцию*

** arg1, arg2, ... - список аргументов, которые нужно передать функции */*

func.bind(context[, arg1[, arg2[, ...]]])

ПРИВЯЗКА КОНТЕКСТА (ПРИМЕР)

```
let mouse = {  
  model: 'M100',  
  getModel: function() {  
    return this.model;  
  }  
}  
  
// сохраним в переменную ссылку на метод  
let getModelMouse = mouse.getModel;  
// вызовем метод getModelMouse и выведем его результат  
console.log(getModelMouse()); //undefined  
/* сохраним в переменную ссылку на метод с привязкой его к  
контексту mouse */  
let getModelMouseBind = mouse.getModel.bind(mouse);  
// вызовем метод getModelMouseBind и выведем его результат  
console.log(getModelMouseBind()); // "M100"
```


ЧТО ПОЧИТАТЬ ДОПОЛНИТЕЛЬНО

<https://learn.javascript.ru/object-methods> - Методы объекта, "this"

<https://learn.javascript.ru/call-apply-decorators> - Декораторы и переадресация вызова, call/apply

<https://learn.javascript.ru/bind> - Привязка контекста к функции

<https://learn.javascript.ru/arrow-functions> - Повторяем стрелочные функции