

# Руководство MotorXP- AFM Scripting API

---

None

*MotorXP Team*

*None*

## Table of contents

---

1. Введение	3
1.1 Описание	3
1.2 Официальный сайт	3
1.3 Контакты	3
2. Глобальные функции	4
2.1 Глобальные функции	4
2.2 include(path)	5
2.3 include(path)	6
2.4 include(path)	7
2.5 include(path)	8
3. Константы	9
3.1 Список констант	9
4. Встроенные объекты	13
4.1 Index	13
4.2 console	14
4.3 motor	22
4.4 stator	28
4.5 statorItem	29
4.6 rotor	30
4.7 rotorItem	31
5. Типы объектов	32
5.1 Index	32
6. Материалы	33
6.1 Index	33
7. UI-виджеты	34
7.1 Index	34

# 1. Введение

---

## 1.1 Описание

---

Данная документация описывает программный интерфейс (API) для написания скриптов в приложении **MotorXP-AFM** и для построения геометрии мотора. Включает подробную информацию об [встроенных объектах](#), [типах создаваемых объектов](#), [материалах](#), [константах](#), [операциях с файлами](#), [методах подключения скриптов](#) и [элементах управления](#) (UI-виджеты).

### Информация

Для написания скриптов используется язык программирования JavaScript со спецификацией **ECMAScript 6**.

## 1.2 Официальный сайт

---

Посетите наш веб-сайт: [motorxp.com](http://motorxp.com)

## 1.3 Контакты

---

Если вы обнаружите какие-либо ошибки или неточности в документации, пожалуйста, сообщите о них по электронной почте: [nika\\_homework@mail.ru](mailto:nika_homework@mail.ru), [info@motorxp.com](mailto:info@motorxp.com).

 12 апреля 2025г. 09:03:34

## 2. Глобальные функции

---

### 2.1 Глобальные функции

---

#### 2.1.1 include(path)

---

##### Description

Включение скрипта

##### Syntax

```
include(path);
```

##### Example

```
1 include('path_to_script');
```

#### 2.1.2 require(path)

---

##### Description

Загрузка модуля

##### Syntax

```
require(path);
```

##### Example

```
1 let m = require(path)
```

🕒 12 апреля 2025г. 07:21:15

## 2.2 include(path)

---

### 2.2.1 Описание

---

### 2.2.2 Возвращаемое значение

---

None

### 2.2.3 Синтаксис

---

### 2.2.4 Параметры

---

- `path` (строка, обязательно): Путь к файлу скрипта.

### 2.2.5 Пример

---

1



См. также

🕒 12 апреля 2025г. 07:21:15

## 2.3 include(path)

---

### 2.3.1 Описание

---

### 2.3.2 Возвращаемое значение

---

None

### 2.3.3 Синтаксис

---

### 2.3.4 Параметры

---

- `path` (строка, обязательно): Путь к файлу скрипта.

### 2.3.5 Пример

---

1



См. также

🕒 12 апреля 2025г. 09:03:34

## 2.4 include(path)

---

### 2.4.1 Описание

---

### 2.4.2 Возвращаемое значение

---

None

### 2.4.3 Синтаксис

---

### 2.4.4 Параметры

---

- `path` (строка, обязательно): Путь к файлу скрипта.

### 2.4.5 Пример

---

```
1
```



См. также

🕒 12 апреля 2025г. 09:03:34

## 2.5 include(path)

---

### 2.5.1 Описание

---

### 2.5.2 Возвращаемое значение

---

None

### 2.5.3 Синтаксис

---

### 2.5.4 Параметры

---

- `path` (строка, обязательно): Путь к файлу скрипта.

### 2.5.5 Пример

---

```
1
```



См. также

🕒 12 апреля 2025г. 09:03:34



## 3. Константы

---

### 3.1 Список констант

---

#### 3.1.1 Motor

---

- `Motor.SR` — статор-ротор.  
Значение: 0
- `Motor.SRS` — статор-ротор-статор.  
Значение: 1
- `Motor.SRSRS` — статор-ротор-статор-ротор-статор.  
Значение: 2
- `Motor.RSR` — ротор-статор-ротор.  
Значение: 3
- `Motor.RSRSR` — ротор-статор-ротор-статор-ротор.  
Значение: 4

#### 3.1.2 Stator

---

- `Stator.Yokeless` — без ярма.  
Значение: 0
- `Stator.Yoke` — с ярмом.  
Значение: 1

#### 3.1.3 StatorItem

---

- `StatorItem.ID1` — элемент 1.  
Значение: 1
- `StatorItem.ID2` — элемент 2.  
Значение: 2
- `StatorItem.ID3` — элемент 3.  
Значение: 3
- `StatorItem.LayerLower` — нижний слой.  
Значение: 1
- `StatorItem.LayerUpper` — верхний слой.  
Значение: 2

#### 3.1.4 Rotor

---

- `Rotor.Yokeless` — без ярма.  
Значение: 1
- `Rotor.Yoke` — с ярмом.  
Значение: 2

#### 3.1.5 RotorItem

---

- `RotorItem.ID1` — элемент 1.  
Значение: 1
- `RotorItem.ID2` — элемент 2.  
Значение: 2

- `RotorItem.ID3` — элемент 3.  
Значение: 3
- `RotorItem.LayerLower` — нижний слой.  
Значение: 1
- `RotorItem.LayerUpper` — верхний слой.  
Значение: 2

### 3.1.6 PoleArrangement

---

- `PoleArrangement.NN` — NN.  
Значение: 0
- `PoleArrangement.NS` — NS.  
Значение: 1
- `PoleArrangement.NSNS` — NSNS.  
Значение: 2
- `PoleArrangement.NSSN` — NSSN.  
Значение: 3
- `PoleArrangement.NNSS` — NNSS.  
Значение: 4
- `PoleArrangement.NNNN` — NNNN.  
Значение: 5

### 3.1.7 Coil

---

- `Coil.CW` — намотка по часовой стрелке.  
Значение: 1
- `Coil.CCW` — намотка против часовой стрелки.  
Значение: 2

### 3.1.8 Winding

---

#### Тип обмотки

- `Winding.Planar` — планарная.  
Значение: 1
- `Winding.Toroidal` — тороидальная.  
Значение: 2

#### Количество слоев

- `Winding.SingleLayer` — одинарный.  
Значение: 1
- `Winding.DoubleLayer` — двойной.  
Значение: 2

#### Ориентация слоев

- `Winding.UpperLower` — верхний-нижний.  
Значение: 1
- `Winding.LeftRight` — левый-правый.  
Значение: 2

**Тип соединения**

- `Winding.Star` — звезда.  
Значение: 1
- `Winding.Delta` — треугольник.  
Значение: 2

**Подключения**

- `'1-2'`
- `'1||2'`
- `'1-2-3'`
- `'1||2||3'`
- `'1-2-3-4'`
- `'1||2||3||4'`
- `'(1-2)|| (3-4)'`
- `'(1-3)|| (2-4)'`
- `'(1-4)|| (2-3)'`

**3.1.9 Magnetization**

---

- `Magnetization.From` — направление "от".  
Значение: 1
- `Magnetization.Toward` — направление "к".  
Значение: 2
- `Magnetization.CW` — по часовой стрелке.  
Значение: 3
- `Magnetization.CCW` — против часовой стрелки.  
Значение: 4

### 3.1.10 Именованные цвета

---

Список доступных цветов и их визуальное представление:

- `Qt.black` — черный  

- `Qt.white` — белый  

- `Qt.darkGray` — темно-серый  

- `Qt.gray` — серый  

- `Qt.lightGray` — светло-серый  

- `Qt.red` — красный  

- `Qt.green` — зеленый  

- `Qt.blue` — синий  

- `Qt.cyan` — голубой  

- `Qt.magenta` — пурпурный  

- `Qt.yellow` — желтый  

- `Qt.darkRed` — темно-красный  

- `Qt.darkGreen` — темно-зеленый  

- `Qt.darkBlue` — темно-синий  

- `Qt.darkCyan` — темно-голубой  

- `Qt.darkMagenta` — темно-пурпурный  

- `Qt.darkYellow` — темно-желтый  


🕒 12 апреля 2025г. 04:13:06

## 4. Встроенные объекты

---

### 4.1 Index

---

🕒 12 апреля 2025г. 09:03:34

## 4.2 console

---

### 4.2.1 Обзор объекта console

---

The `console` object provides methods for logging information, warnings, errors, and debugging data to the MotorXP apps console. It is a powerful tool for developers to monitor and debug their scripts.

#### Key Features

- Log messages with different levels ( `log`, `info`, `warn`, `error` ).
- Clear the console output ( `clear` ).
- Inspect objects and their properties ( `dir` ).

For detailed information about each method, see the [methods documentation](#).

🕒 12 апреля 2025г. 06:15:03

## 4.2.2 methods

---

### Методы объекта console

Below is the list of available methods provided by the `console` object. Click on a method name to see its detailed description and examples.

- `console.log()` : Outputs a message to the console. Accepts any number of arguments of any type.
- `console.info()` : Outputs an informational message to the console. Similar to `log`, but intended for informational purposes.
- `console.warn()` : Outputs a warning message to the console. Used to indicate potential issues that are not critical but should be addressed.
- `console.error()` : Outputs an error message to the console. Used to indicate critical issues that need immediate attention.
- `console.clear()` : Clears the console output, removing all previously logged messages.
- `console.dir()` : Displays an interactive list of the properties of a specified JavaScript object. Useful for inspecting the structure of objects.

 12 апреля 2025г. 06:15:03

## console.log()

### ОПИСАНИЕ

Метод `console.log()` выводит сообщение серым цветом в консоль. Он может принимать несколько аргументов и отображать их в одной строке.

### СИНТАКСИС

```
console.log(message1 : any, message2 : any, ..., messageN : any);
```

### ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

None

### ПРИМЕР

```
1 // Один аргумент
2 console.log('log text');
3
4 // Несколько аргументов
5 console.log('text1', 'text2');
6 console.log('text1', 'text2', .. , 'text10');
7
8 // Различные типы аргументов
9 console.log('text1', 123, true, { key: 'value' });
```



См. также

[console.info\(\)](#)

[console.warn\(\)](#)

[console.error\(\)](#)

🕒 12 апреля 2025г. 07:20:48



## console.info()

### ОПИСАНИЕ

The `console.info()` method outputs an informational message to the console. It is similar to `console.log()`, but it is intended for informational purposes and may be styled differently in some environments (e.g., with an "info" icon or color).

### СИНТАКСИС

```
console.info(message1 : any, message2 : any, ..., messageN : any);
```

### ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

None

### ПРИМЕР

```
1 // Single argument
2 console.info('info text');
3
4 // Multiple arguments
5 console.info('text1', 'text2');
6 console.info('text1', 'text2', .. , 'text10');
7
8 // Mixed types
9 console.info('text1', 123, true, { key: 'value' });
```

### See also

[console.info\(\)](#)

[console.warn\(\)](#)

[console.error\(\)](#)

 12 апреля 2025г. 07:20:48

## console.warn()

### ОПИСАНИЕ

The `console.warn()` method outputs a warning message to the console. It is used to indicate potential issues that are not critical but should be addressed. Warning messages are often styled differently in the console (e.g., with a yellow background or an alert icon).

### СИНТАКСИС

```
console.warn(message1 : any, message2 : any, ..., messageN : any);
```

### ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

None

### ПРИМЕР

```
1 // Single argument
2 console.warn('warn text');
3
4 // Multiple arguments
5 console.warn('text1', 'text2');
6 console.warn('text1', 'text2', .. , 'text10');
7
8 // Mixed types
9 console.warn('text1', 123, true, { key: 'value' });
```

🕒 12 апреля 2025г. 07:20:48

## console.error()

### ОПИСАНИЕ

The `console.error()` method outputs an error message to the console. It is used to indicate critical issues that need immediate attention. Error messages are often styled differently in the console (e.g., with a red background or an error icon).

### СИНТАКСИС

```
console.error(message1 : any, message2 : any, ..., messageN : any);
```

### ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

`None`

### ПРИМЕР

```
1 // Single argument
2 console.error('critical text');
3
4 // Multiple arguments
5 console.error('text1', 'text2');
6 console.error('text1', 'text2', .. , 'text10');
7
8 // Mixed types
9 console.error('text1', 123, true, { key: 'value' });
```

🕒 12 апреля 2025г. 07:20:48

## **console.clear()**

### ОПИСАНИЕ

The `console.clear()` method clears the console output, removing all previously logged messages. This is useful for resetting the console and ensuring a clean workspace for debugging or logging new information.

### СИНТАКСИС

```
console.clear();
```

### ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

`None`

🕒 12 апреля 2025г. 06:15:03

## console.dir()

### ОПИСАНИЕ

The `console.dir()` method displays an interactive list of the properties of a specified JavaScript object. It is particularly useful for inspecting the structure and contents of objects, as it formats the output in a collapsible tree-like structure that makes it easier to explore nested properties.

Unlike `console.log()`, which may display objects in a simplified or stringified format, `console.dir()` provides a detailed and interactive view of the object's properties.

### СИНТАКСИС

```
console.dir(object : any);
```

### ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

None

### ПРИМЕР

```
1 // Define an object
2 const obj = {
3   name: 'MotorXP',
4   version: 1.0,
5   features: ['AFM Design', 'Electromagnetic Analysis', 'Optimization API'],
6   details: {
7     developer: 'MotorXP Team',
8     license: 'Combo'
9   }
10 };
11
12 // Inspect the object
13 console.dir(obj);
```

#### Output

```
► Object
  name: "MotorXP"
  version: 1
  ► features: Array(3)
    0: "AFM Design"
    1: "Electromagnetic Analysis"
    2: "Optimization API"
  ► details: Object
    developer: "MotorXP Team"
    license: "Combo"
```

🕒 12 апреля 2025г. 07:20:48

## 4.3 motor

---

### 4.3.1 Motor Overview

---

#### Description

Встроенный объект `Motor` представляет собой модель двигателя (генератора) с различными параметрами.

🕒 11 апреля 2025г. 16:00:15

## 4.3.2 properties

---

### Motor Properties

PROPERTIES OF THE `motor` OBJECT

- `machineType` : Type of the machine.
- `height` : Height of the machine.
- `periodicity` : Periodicity.
- `periodicityAngleSpan` : Angular span of the periodicity.
- `boundaryConditionMode` : Mode of the boundary conditions.
- `boundaryCondition` : Boundary condition applied to the motor.
- `simDomain` : Simulation domain of the motor (READONLY, returns `SimDomain` ).
- `simRadialOuterDomain` : Radial outer domain of the simulation (READONLY).
- `stator` : Returns the stator object associated with the motor.
- `rotor` : Returns the rotor object associated with the motor.
- `winding` : Returns the winding object associated with the motor.
- `mesh` : Returns the mesh object associated with the motor.

🕒 11 апреля 2025г. 16:00:15

## machineType

### ОПИСАНИЕ

The `machineType` property defines the configuration type of the motor.

### ПРИНИМАЕМЫЕ ЗНАЧЕНИЯ:

- `Motor.SR` (0): stator - rotor
- `Motor.SRS` (1): stator - rotor - stator
- `Motor.SRSRS` (2): stator - rotor - stator - rotor - stator
- `Motor.RSR` (3): rotor - stator - rotor
- `Motor.RSRSR` (4): rotor - stator - rotor - stator - rotor

### ТИП ЗНАЧЕНИЯ СВОЙСТВА:

Number

### ДОСТУП

Чтение\Запись

### СИНТАКСИС

```
motor.machineType = Motor.<TYPE>;  
motor.machineType;
```

### ПРИМЕР

```
1 // Set the machine type to Stator-Rotor-Stator configuration  
2 motor.machineType = Motor.SRS;  
3  
4 // Check the current machine type  
5 console.log(motor.machineType); // Output: 1
```



См. также

🕒 12 апреля 2025г. 09:03:34



**objectName**

## ОПИСАНИЕ

`objectName`

## ВОЗМОЖНЫЕ ЗНАЧЕНИЯ:

- `Motor.SR (0)`: stator - rotor

## СИНТАКСИС

## ДОСТУП

`Чтение\Запись`

## ПРИМЕР

`1` **См. также** 12 апреля 2025г. 09:03:34

## height

### ОПИСАНИЕ

Свойство `height` определяет высоту машины.

### Пример:

```
motor.height = 100;
```

🕒 11 апреля 2025г. 16:00:15

## periodicity

### ОПИСАНИЕ

Свойство `periodicity` определяет периодичность машины.

### Пример:

```
motor.periodicity = 100;
```

🕒 11 апреля 2025г. 16:00:15

## 4.4 stator

---

### 4.4.1 Motor Overview

---

#### Description

Встроенный объект `Motor` представляет собой модель двигателя (генератора) с различными параметрами.

🕒 11 апреля 2025г. 16:00:15

## 4.5 statorItem

---

### 4.5.1 Motor Overview

---

**Description**

Встроенный объект `Motor` представляет собой модель двигателя (генератора) с различными параметрами.

 11 апреля 2025г. 16:00:15

## 4.6 rotor

---

### 4.6.1 Motor Overview

---

#### Description

Встроенный объект `Motor` представляет собой модель двигателя (генератора) с различными параметрами.

🕒 11 апреля 2025г. 16:00:15

## 4.7 rotorItem

---

### 4.7.1 Motor Overview

---

#### Description

Встроенный объект `Motor` представляет собой модель двигателя (генератора) с различными параметрами.

🕒 11 апреля 2025г. 16:00:15

## 5. Типы объектов

---

### 5.1 Index

---

🕒 12 апреля 2025г. 09:03:34



## 6. Материалы

---

### 6.1 Index

---

🕒 12 апреля 2025г. 09:03:34

## 7. UI-виджеты

---

### 7.1 Index

---

🕒 12 апреля 2025г. 09:03:34