Каждое значение в JavaScript имеет свой тип. А функция – это какой тип?

В JavaScript функции – это объекты.

Можно представить функцию как «объект, который может делать какое-то действие». Функции можно не только вызывать, но и использовать их как обычные объекты: добавлять/удалять свойства, передавать их по ссылке и т.д.

**[Свойство «name»](https://learn.javascript.ru/function-object" \l "svoystvo-name)**

имя функции нам доступно как свойство «name»:

function sayHi() {

alert("Hi");

}

alert(sayHi.name); // sayHi

name присваивает корректное имя даже в случае, когда функция создаётся без имени и тут же присваивается, вот так:

let sayHi = function() {

alert("Hi");

};

alert(sayHi.name); // sayHi (есть имя!)

не могу понять пример ниже

Это работает даже в случае присваивания значения по умолчанию:

function f(sayHi = function() {}) {

alert(sayHi.name); // sayHi (работает!)

}

f();

В спецификации это называется «контекстное имя»: если функция не имеет name, то JavaScript пытается определить его из контекста.

Также имена имеют и методы объекта:

let user = {

sayHi() {

// ...

},

sayBye: function() {

// ...

}

}

alert(user.sayHi.name); // sayHi

alert(user.sayBye.name); // sayBye

Бывает, что корректное имя определить невозможно. В таких случаях свойство name имеет пустое значение. Например:

// функция объявлена внутри массива

let arr = [function() {}];

alert( arr[0].name ); // <пустая строка>

// здесь отсутствует возможность определить имя, поэтому его нет

Впрочем, на практике такое бывает редко, обычно функции имеют name.

## [Свойство «length»](https://learn.javascript.ru/function-object" \l "svoystvo-length)

Ещё одно встроенное свойство «length» содержит количество параметров функции в её объявлении.

function f1(a) {}

function f2(a, b) {}

function many(a, b, ...more) {}

alert(f1.length); // 1

alert(f2.length); // 2

alert(many.length); // 2

Как мы видим, троеточие, обозначающее «остаточные параметры», здесь как бы «не считается»

Свойство length иногда используется для [интроспекций](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) в функциях, которые работают с другими функциями.

function ask(question, ...handlers) {

let isYes = confirm(question);

for(let handler of handlers) {

if (handler.length == 0) {

if (isYes) handler();

} else {

handler(isYes);

}

}

}

// для положительных ответов вызываются оба типа обработчиков

// для отрицательных - только второго типа

ask("Вопрос?", () => alert('Вы ответили да'), result => alert(result));

функция ask принимает в качестве параметров вопрос question и произвольное количество функций-обработчиков ответа handler.

Когда пользователь отвечает на вопрос, функция вызывает обработчики. Мы можем передать два типа обработчиков:

* Функцию без аргументов, которая будет вызываться только в случае положительного ответа.
* Функцию с аргументами, которая будет вызываться в обоих случаях и возвращать ответ.

Чтобы вызвать обработчик handler правильно, будем проверять свойство handler.length.

Идея состоит в том, чтобы иметь простой синтаксис обработчика без аргументов для положительных ответов (наиболее распространённый случай), но также и возможность передавать универсальные обработчики