**Вопросы 💎**

1. Какие есть способы объявления функций? Приведите примеры вызова одной и той же функции всеми известными вам способами.

There are several ways of declaring functions. The two most obvious ones are

Function Expression and Function Declaration.

Function Declaration:

function sum(a, b) {

return a + b;

}

Function Expression:

var sum = function(a, b) {

return a + b;

};

Based on the above examples, in the case of Function Declaration, the interpreter declares a variable with the function name and assigns function description.

In the case of Function Expression - we ourselves declare a variable with the function name and assign a function description to it. In this case, we can function name. In other words: If the function declaration is part of some other expression, then this declaration is called Function Expression. In this case, the function declaration is part of an expression by `sum` variable declaration.

Immediately-invoked function expression is a type of function call where function is called immediately after it is declared. The following pattern is used for this:

(function(parameters) {code

function})(arguments);

Example:

(function(a, b) {

console.log(a + b);

})(1, 1);

In this case, the function is declared without a name and immediately called with 1 and 1 arguments.

There is an even simpler and shorter syntax for creating functions, which is often better than the Function Expression syntax.

It's called "arrow functions" because it looks like this:

let func = (arg1, argN) => expression

Such code creates a `func` function with arguments `arg1..argN` and computes `expression` on the right hand side using them, returning the result.

In other words, this is a shorter version of such an entry:

let func = function(arg1, argN) {

return expression;

};

Let's take a look at a concrete example:

let sum = (a, b) => a + b;

/\* A shorter form for:

let sum = function(a, b) {

return a + b;

};

\*/

alert( sum(1, 2) ); // 3

That is, `(a, b) => a + b` specifies a function with two arguments `a` and `b` that, when run, evaluates the expression to the right `a + b` and returns its result.

- If we have only one argument, the parentheses around the parameters can be omitted, making the notation even shorter:

// the same as

// let double = function(n) { return n \* 2 }

let double = n => n \* 2;

alert(double(3)); // 6

- If there are no arguments, empty parentheses are given:

let sayHi = () => alert("Hello!");

sayHi();

1. В чем разница между тестированием и отладкой (дебаггингом)? А что такое логирование?

Testing is the action of checking whether the actual results correspond to the expected results of the software and making sure that it is free from defects. Debugging is the process of finding and fixing defects or problems in a computer program that prevent the software or system from working properly.

The main difference between testing and debugging is that testing is the process of finding and locating defects in the software, whereas debugging is the process of removing the detected defects.

Also, testing is performed by the testing team, while debugging is performed by the development team.

Another difference between testing and debugging is purpose. The goal of testing is to find as many defects as possible. The goal of debugging, however, is to fix the defects found.

While testing can be done manually or automatically, debugging is done manually.

Logging is the recording of logs. It helps to answer questions about what happened, when and under what circumstances. Without logging, it is difficult to understand why an error occurs, if it occurs periodically and only under certain conditions. To make it easier for administrators and programmers, the log contains not only information about the errors, but also the reasons for them.

Once in production, the performance of the application needs to be continuously monitored to prevent and quickly respond to potential upsets. Log analysis is one of the basic tools for IT professionals. It helps to discover the source of many problems, identify conflicts in configuration files, and track security events. And, most importantly, logging allows errors to be corrected quickly. This is why logging is so important when debugging programs, finding the source of problems with application software and databases.

Here are some typical cases in which logging is used:

An administrator looks for the causes of technical problems, device or operating system failures, and website inaccessibility.

A developer debugging, i.e. looking for, locating and resolving errors.

If something needs to be output to the console from the code, the console.log function is used.

If you build the logging in your application correctly, you can figure out what's going on in the code without the debugger.

1. В чем разница между Function Expression и Function Declaration?

First, the syntax: how to determine what is what in the code.

- Function Declaration: a function is declared by a separate construct „function...'' in the main code stream.

function sum(a, b) {

return a + b;

}

- Function Expression: A function created inside another expression or syntax construct. In this case, the function is created in the right side of an „assignment expression'' =':

let sum = function(a, b) {

return a + b;

};

A more subtle difference is *when* the function is created by the JavaScript engine.

Function Expression is created when execution reaches it, and can then be used.

Once the execution thread has reached the right side of the assignment expression `let sum = function...` - at this point, the function is considered created and can be used (assigned to a variable, called, etc. ).

Things are different with Function Declaration.

Function Declaration can be used in the whole script (or code block, if the function is declared in a block).

In other words, when the JavaScript engine prepares to execute a script or code block, it first looks within it for a Function Declaration and creates all such functions. You can think of this process as the "initialization phase".

Only after all the Function Declarations have been processed will execution continue.

As a result, functions created as Function Declarations can be called before their definitions.

For example, it would work like this:

sayHi("Vasya"); // hello, Vasya

function sayHi(name) {

alert( ``Hi, ${name}'' );

}

The SayHi function was created when the JavaScript engine was preparing the script for execution, and such a function is visible everywhere in this script.

If it was a Function Expression, such code would cause an error:

sayHi("Vasya"); // error!

let sayHi = function(name) { // (\*) no more magic

alert( ``Hi, ${name}'' );

};

1. Что делает функция console.log()?

If something needs to be output to the console from the code, the `console.log` function is used.

For example, bring value 5 to the console:

// Open console first to see the result

console.log(5);

An ordinary website user will not see this output, because it is in the console. Recall that the console can be opened via the developer tools.

If you build the logging in the application correctly, you can figure out what's going on in the code even without a debugger.

1. Что такое BOM и DOM?

Document Object Model, DOM for short, is an object model of a document which represents all the content of a page as objects which can be changed.

In DOM every HTML tag is an object. Nested tags are the "children" of the parent element. Text inside a tag is also an object.

All these objects are accessible by JavaScript, we can use them to modify the page.

The `document` object is the main "entry point". We can use it to create or change something on the page.

The Browser Object Model (BOM) is the additional objects provided by the browser (environment) to handle everything but the document.

For example:

- The [navigator] object gives information about the browser itself and the operating system. Among its many properties, the most famous are: `navigator.userAgent` - information about the current browser, and `navigator.platform` - information about the platform (may help to understand what operating system the browser is open in - Windows/Linux/Mac and so on).

- The [location] object allows you to retrieve the current URL and redirect the browser to the new address.

1. Есть вот такая страница:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Как найти в ней?…

* 1. Таблицу с id="age-table"

let ageTable = document.getElementById('age-table');

console.log(ageTable)

* 1. Все элементы label внутри этой таблицы (их три)

let labels = table.getElementsByTagName('label');

console.log(labels);

* 1. Форму form с именем name="search"

let search = document.getElementsByTagName('form').getElementsByName('search';

console.log(search);

1. Как сделать переход на другую страницу при клике на кнопку (без <a href=...>, только средствами JavaScript)?

<input id="search" type="button" value="Search!" onclick="window.location = 'https://www.yandex.ru';">

Or alternatively:

<input id="search" type="button" value="Search!" onclick="func()">

function func() {

    document.location.href = "https://www.yandex.ru";

}

1. Как можно обнулить (очистить) значение внутри input?

document.getElementsByTagName('input').reset();

1. Что такое *this*?

The JavaScript this keyword refers to the object it belongs to.

It has a specific meaning depending on the context in which it is applied.

In a method, this refers to the owner object.

Alone, this refers to the global object.

In a function, this refers to the global object.

In a function, in strict mode, this is undefined.

In an event, this refers to the element that received the event.

Methods like call(), and apply() can refer this to any object.

fullName : function() {  
  return **this**.firstName + " " + **this**.lastName;  
}

1. Как будет выглядеть ваша функция приветствия из прошлого домашнего задания, если ее переписать в стрелочном формате?

let showMessage = () => {

let name = prompt('What is your name?', "Kate");

alert(`Hello, ${name}!`);

}