Основи програмування

Заняття #6

LOGOS
IT ACADEMY

- Function Expression
- Arrow-function
- Recursion

Синтаксис Function Expression

```
let myFunction = function [name](paramN) {
   statements
};
```

name

Ім'я функції. Можна пропустити, в цьому випадку функція буде анонімною. Ім'я доступне лише у тілі функції.

paramN

Ім'я аргументу, який передається у функцію.

statements

Інструкції, що складають тіло функції.

LOGOS IT ACADEMY

Function Expression

Функціональний вираз дуже схожий на оголошення функції та має майже такий самий синтаксис (дивіться оголошення функції).

Головна відмінність між функціональним виразом та оголошенням функції - це ім'я функції, яке може бути пропущене у функціональних виразах для створення анонімних функцій.

Функціональний вираз можна використовувати як НВФВ (Негайно виконуваний функціональний вираз), який запускається одразу після визначення.

LOGOS IT ACADEMY

Підняття функціонального виразу

// Функціональні вирази(function Expression) у JavaScript не піднімаються, на відміну від оголошень функцій(function Declaration). Не можна використовувати функціональний вираз до його визначення:

```
console.log(notHoisted) // undefined
//хоча імена змінних піднімаються, визначення не піднімається і дорівнює undefined.
notHoisted(); // TypeError: notHoisted is not a function

let notHoisted = function() {
   console.log('bar');
};
// У Оголошення функцій у JavaScript піднімаються наверх замикаючої функції або глобальної області видимості. Ви можете використовувати функцію до того, як оголосили її:
hoisted(); // виводить "foo"

function hoisted() {
  console.log('foo');
}
```

LOGOS IT ACADEMY

Function Expression

```
// Виводить Hello JS в console.log.

let sayHello = function(){
        console.log(`Hello JS`)
}
sayHello();

// Функція з 1 параметром, яка виведе Hello Ivan в console.log.

let sayHello = function(a){ // a – параметр функції
        console.log(`Hello ${a}`);
}
sayHello('Ivan'); // передаємо значення в параметр функції.
```

LOGOS IT ACADEMY

Function Expression

```
// Функція з 2 параметрами, яка виведе Hello Ivan Ivanov в console.log.

let sayHello = function(a,b){ // a,b – параметри функції
        console.log(`Hello ${a} ${b}`);
}
sayHello('Ivan', 'Ivanov'); // передаємо значення в параметри фунції.

// Якщо у функції є параметри, а ми нічого не передаємо, то параметр буде мати значення undefined

let sayHello = function(a,b){
        console.log(`Hello ${a} ${b}`);
}
sayHello(); // Функція виведе повідомлення в console.log Hello undefined undefined
```

Function Declaration

```
// Параметри функції можна задавати значеннями за замовчуванням(старіший варіант)
let sayHello = function(myName){
        myName = myName || 'guest';
        console.log(`Hello ${myName}`);
}
sayHello('Petro'); // Функція виведе повідомлення в console.log Hello Petro
sayHello(); // Функція виведе повідомлення в console.log Hello undefined

// Параметри функції можна задавати значеннями за замовчуванням(новіший варіант)
let sayHello = function(myName = 'guest'){
        console.log(`Hello ${myName}`);
}
sayHello('Petro'); // Функція виведе повідомлення в console.log Hello Petro
sayHello(); // Функція виведе повідомлення в console.log Hello undefined
```

LOGOS IT ACADEMY

Function Declaration

```
// За допомогою інструкції return функція може повернути деяке значення (результат роботи функції) програмою, яка її викликала. Значення, що повертається передається в точку виклику функції.
let sayHello = function(myName){
        return `Hello ${myName}`;
}
let message = sayHello('Petro'); // Функція поверне повідомлення Hello Petro і збереже в змінну message
console.log(message); // Виводимо значення в console.log

// Інструкція return може бути розташована в будь-якому місці функції. Як тільки буде досягнута інструкція return, функція повертає значення і негайно завершує своє виконання. Код, розташований після інструкції return, буде проігнорований:
let sayHello = function(myName){
        return `Hello ${myName}`;
        console.log('Hello'); // Цей console.log не буде виконуватися
}
```

Синтаксис Arrow function

(param1, param2, …, paramN) => { statements }

(param1, param2, …, paramN) => expression

// еквівалентно до: => { return expression; }

(singleParam) => { statements }

singleParam => { statements }

// Якщо у функції тільки один параметр, то дужки не обов'язкові:

() => { statements }

// Список параметрів для функції без параметрів повинен бути записаний у парі фігурних дужок.

paramN

Ім'я аргументу, який передається у функцію.

statements

Інструкції, що складають тіло функції.

LOGOS IT ACADEMY

Arrow function

Однією з найпомітніших нововведень сучасного JavaScript стала поява стрілочних функцій (arrow function).

При оголошенні таких функцій використовують особливу комбінацію символів - =>.

У стрілочних функцій є дві основні переваги перед традиційними функціями. Перше - це дуже зручний і компактний синтаксис.

Друге полягає в тому, що підхід до роботи зі значенням this в стрілочних функціях виглядає інтуїтивно зрозуміліше, ніж в звичайних функціях.

LOGOS IT ACADEMY

Arrow function

```
// Стрілочна ф-я без параметрів і без return. Виводить Hello JS в console.log.

let sayHello = () => {
        console.log(`Hello JS`)
}
sayHello();

// Стрілочна ф-я з 1 параметром і без return, яка виведе Hello Ivan в console.log.

let sayHello = (a) => { // a – параметр функції
        console.log(`Hello ${a}`);
}
sayHello('Ivan'); // передаємо значення в параметр фунції.
```

LOGOS IT ACADEMY

Arrow function

```
// Стрілочна ф-я з 2 параметрами і без return, яка виведе Hello Ivan Ivanov в console.log.

let sayHello = (a,b) => { // a,b – параметри функції
        console.log(`Hello ${a} ${b}`);
}
sayHello('Ivan', 'Ivanov'); // передаємо значення в параметри фунції.

// Якщо у функції є параметри, а ми нічого не передаємо, то параметр буде мати значення undefined

let sayHello = (a,b) => {
        console.log(`Hello ${a} ${b}`);
}
sayHello(); // Функція виведе повідомлення в console.log Hello undefined undefined
```

LOGOS IT ACADEMY

Arrow function

```
// Стрілочна ф-я з 2 параметрами і з return
let sayHello = (a,b) => `Hello ${a}, ${b}`;

let message = sayHello('Petro', 'Petriv'); // Функція поверне повідомлення Hello Petro Petriv і збереже в змінну message
console.log(message); // Виводимо значення в console.log

// Стрілочна ф-я з 2 параметрами і з return. Явне вказування return
let sayHello = (a,b) => {
        let message = `Hello ${a} ${b}`
    return message;
}
```

LOGOS IT ACADEMY

Arrow function

```
// Стрілочна ф-я з 1 параметрамом і з return. Якщо у функції тільки один параметр, то дужки не обов'язкові:
let sayHello = a => `Hello ${a}`;

let message = sayHello('Petro'); // Функція поверне повідомлення Hello Petro і збереже в змінну message
console.log(message); // Виводимо значення в console.log
```

Recursion

```
// Приклад обчислення додатньої степіні через рекурсію

function f(a, b) {
        return (b != 0) ? a * f(a, b - 1) : 1;
}
let a = prompt("Enter number");//5
let b = prompt("Enter pow");//3
console.log(f(a, b));
```

Рекурсія — це виклик функцією самої себе.

Рекурсію застосовують, коли розв'язувана задача містить подібні до себе підзадачі.

Залежно від вхідних даних розрізняють кінцевий виклик (розв'язує найпростішу задачу) та проміжний (має підзадачі, тож передбачає щонайменше один рекурсивний виклик).

LOGOS IT ACADEMY

Recursion

Виконання програми багаторазово спускається вниз, поки не упреться в умова виходу з рекурсії. Досягнувши кінця, вона йде назад, повертаючи результати зроблених викликів.

Рекурсивна функція обов'язково повинна мати умова завершення, якщо його не вказати, функція буде викликатися до тих пір,поки не буде досягнута максимальна глибина рекурсії, потім буде згенеровано виняток.

Загальна кількість вкладених викликів називають глибиною рекурсії.

Максимальна глибина рекурсії в браузерах обмежена 10 000 рекурсивних викликів.

Додаткові джерела інформації:

Посилання

https://learn.javascript.ru/arrow-functions-basics
https://learn.javascript.ru/recursion
https://learn.javascript.ru/function-basics
https://learn.javascript.ru/function-declaration-expression
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Functions/Arrow_functions
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Statements/function
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/function
http://яваскрипт.укр/function
http://яваскрипт.укр/arguments